


ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO EN INFORMÁTICA

Título del proyecto:

“RECORRIDO INTERACTIVO DEL PARQUE DE TRINITARIOS DE PAMPLONA MEDIANTE DISPOSITIVOS MÓVILES”

Julen San Emeterio Uriz

Alfredo Pina Calafi

Pamplona, 30 de junio de 2015

# ÍNDICE

# 1 INTRODUCCIÓN

En este proyecto se ha creado un recorrido interactivo a lo largo del parque de Trinitarios de Pamplona realizable mediante el uso de dispositivos móviles. El proyecto surge de una colaboración entre la UPNA y Opera Ingeniería, empresa que proyectó el parque, con la idea de dotar al mismo de un valor añadido y de ofrecer al visitante una forma diferente de pasear y de interactuar con el entorno.

Con este propósito, se ha recopilado información acerca de los árboles más significativos del parque, que al mismo tiempo son una pequeña muestra del patrimonio natural de toda Navarra. Esta información de cada una de las especies incluye tanto datos que pueden resultar más enciclopédicos (altura, origen, frutos, usos, etc.), como unos textos asociados a cada árbol, que se han creado expresamente para este proyecto y que cuentan historias, leyendas, cuentos o curiosidades relacionadas con cada uno de ellos.

Para la realización de este proyecto se ha optado por seguir un modelo cliente-servidor, de manera que la información recopilada acerca de cada árbol (texto, imágenes, audio) estará almacenada en la parte del servidor y los distintos programas que actúen como la parte cliente serán los que soliciten esa información para mostrarla al usuario final.

Los programas clientes que mostrarán la información almacenada en el servidor son dos:

- **Aplicación para móviles Android**:
  Se ha desarrollado una aplicación que permite la visualización de toda la información recopilada, de una forma práctica y amigable. Además, esta aplicación sirve para leer los códigos QR presentes en las placas informativas de cada árbol y contiene una audio-guía con localización por GPS que permite recorrer el parque escuchando pequeñas historias de cada árbol a medida que pasas junto a ellos. Esta aplicación ha sido desarrollada en el lenguaje de programación JAVA mediante el IDE Android Studio. Es totalmente gratuita y está disponible en Google Play.

- **Sitio web:**
  Como complemento a la aplicación, se ha creado un sitio web alojado en el mismo hosting que la base de datos. Este sitio web sirve, por una parte, para que aquellos usuarios que no dispongan de un dispositivo Android, puedan al menos escanear los códigos QR con cualquier lector y de esa manera recibir la información de cada árbol en su dispositivo. Por otra parte, este sitio web sirve como puerta de entrada al parque para aquellos que no lo conozcan, y como página de aterrizaje o “landing page” para aquellos usuarios que no conozcan nuestra aplicación y quieran descargarla. El sitio web ha sido desarrollado utilizando los lenguajes HTML5, CSS3 y PHP.

## 1.1 MOTIVACIÓN

La forma en la que nos relacionamos con el mundo que nos rodea no es la misma que hace unos años. El ritmo de vida que llevamos, la falta de tiempo y, sobre todo, la dependencia excesiva de los dispositivos móviles, son algunas de las razones de que muchas veces no prestemos atención al camino por el que nos movemos, a las personas que nos cruzamos o a las vistas que tenemos a nuestro alrededor. Esta circunstancia es más notable aún en las ciudades, donde la gente parece recorrer las calles y parques sin disfrutar de ellas, con prisas, sin ser consciente de la historia, la cultura, la mitología y el arte que puede esconderse en una de sus esquinas o en uno de sus árboles.

Este proyecto nace con la idea de que el uso de los dispositivos móviles deje de ser un obstáculo para convertirse en un estímulo y en una vía de comunicación y fuente de conocimiento en torno al patrimonio cultural y natural de nuestra tierra. Se pretende que, aprovechando las posibilidades que nos ofrece la tecnología, este conocimiento llegue más fácilmente a todo tipo de personas y, en especial, a las nuevas generaciones. Esta idea se engloba en el concepto de ciudad inteligente, buscando hacer del parque un espacio abierto al ciudadano, en el que sus visitantes puedan ser conscientes del entorno, interactuar con él y recibir información acerca de aquello que lo compone. De esa manera será más fácil que las personas tomen conciencia de los árboles, del valor e importancia que éstos han tenido, tienen y tendrán para nosotros, y por lo tanto de la necesidad de cuidarlos y protegerlos.

Así pues, en colaboración Opera Ingeniería, surge la propuesta de crear un recorrido interactivo a lo largo de un parque proyectado por esta empresa, el parque de Trinitarios de Pamplona, situado en el barrio de San Jorge. Este recorrido interactivo se podrá realizar a diferentes niveles, ofreciendo al usuario la libertad de elegir cómo lo quiere recorrer y haciendo que un simple paseo por el parque pueda aportarnos, gracias a la tecnología, un contacto más cercano con la naturaleza y podamos, al mismo tiempo, aprender curiosidades, leyendas e historias asociadas a los elementos que allí podemos encontrar.

## 1.2 EL PARQUE DE TRINITARIOS

Como sabemos, Navarra es una comunidad con un gran patrimonio natural y sus ciudades, en especial Pamplona, es conocida por disponer de amplias zonas verdes, parques y jardines. Entre los numerosos parques de la ciudad podemos encontrar el llamado Parque Fluvial del Arga, un verdadero pulmón verde que ocupa alrededor de un millón de metros cuadrados y que discurre a lo largo de 11 kilómetros siguiendo el curso del río Arga. A través de este parque podemos recorrer y disfrutar de la presencia de puentes históricos, molinos, presas, etc., además de disfrutar de las vistas que de la parte alta de la ciudad y sus murallas se nos presentan.

Dentro de este parque fluvial y formando parte de él encontramos el parque de Trinitarios, situado en la margen izquierda del río y que en total ocupa 150.000 metros cuadrados. El parque se inauguró el pasado año 2014 y es en este parque donde se ha realizado este proyecto.

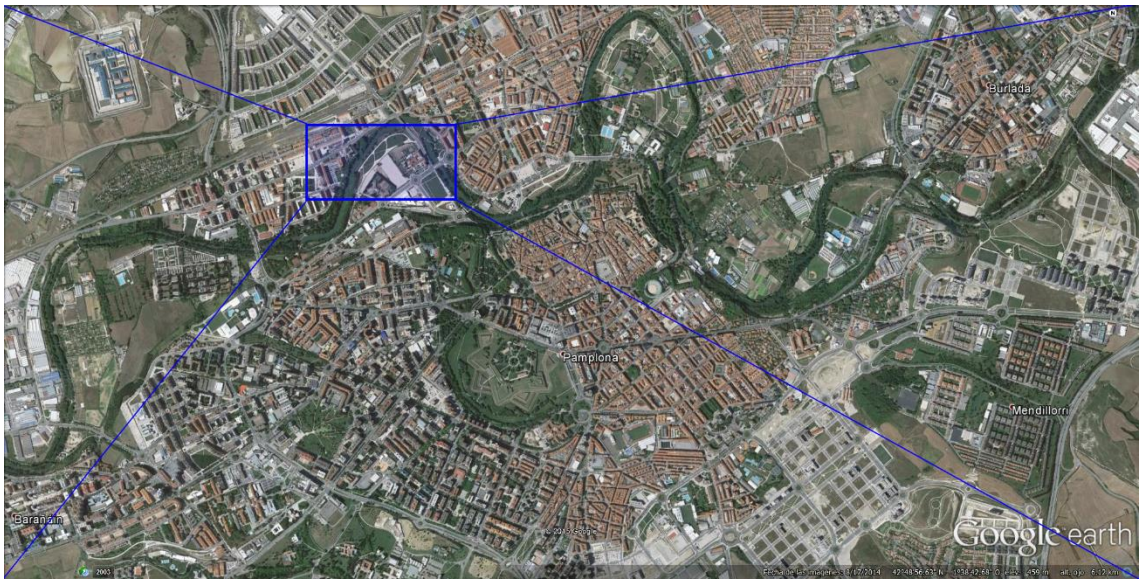



**Figura 1.1 Localización del parque de Trinitarios en Pamplona**


**Figura 1.2 Visión de conjunto del parque de Trinitarios**

Una de las características del parque es que dispone de un arbolado muy variado que representa la generalidad del arbolado navarro. A lo largo del parque encontramos todo tipo de árboles: la mayoría plantados al construir el parque y algunos de ellos anteriores a la construcción y que han sido respetados e incorporados al propio parque.

Con este recorrido se pretende pues dar a conocer las especies vegetales presentes en el parque y por ende en toda Navarra, dotando así al parque de un componente interactivo y al mismo tiempo educativo, que sea atractivo para todo tipo de usuarios y sirva como modelo para otro tipo de parques, zonas de interés o destinos turísticos.

## 1.3 SMART CITIES

El concepto de smart city o ciudad inteligente es un concepto que, a pesar de estar en plena actualidad y ser algo de lo que oímos hablar muy a menudo, sigue resultando algo difuso y que abarca muchos aspectos. Muchas veces el hablar de ciudad inteligente está asociado a una ciudad eficiente energéticamente, que hace uso de las tecnologías de la información y la comunicación (TIC) para recabar datos del consumo energético de la ciudad y gestionarlo de manera que se reduzca en la manera de lo posible. Otras veces lo asociamos a que su red de transporte público está totalmente informatizada y controlada, de manera que los ciudadanos saben con exactitud a qué hora va a llegar su autobús o su metro. Incluso podemos pensar en una ciudad plagada de sensores y cámaras que controlan todos los movimientos y avisan inmediatamente cuando se produce alguna amenaza. Todos estos ejemplos tienen algo en común, y nos acercan a una posible definición de ciudad inteligente: una ciudad que hace uso de las TIC, recopilando y haciendo uso de toda la información posible, para convertirse en una ciudad más habitable y conectada con sus ciudadanos.

Con la proliferación del uso de dispositivos móviles con conexión a internet, esta capacidad de comunicación entre la ciudad y sus habitantes es más sencilla que nunca. Por una parte, los propios dispositivos pueden servir como fuente de información hacia la propia ciudad, ya sea recibiendo un mensaje de un ciudadano, recibiendo datos a través de una aplicación, recibiendo la localización física del dispositivo, etc. Por otra parte, es la ciudad la que comunica de una forma inmediata al ciudadano acerca de aquello que éste necesite saber: cuándo pasa el siguiente autobús, si viene muy lleno, dónde puede encontrar un cajero automático, etc. Al mismo tiempo, el propio dispositivo (ya sea un móvil, un reloj, un chip en nuestro cuerpo) nos puede servir para pagar automáticamente en los establecimientos, registrar nuestras recetas médicas, gestionar el papeleo con la administración automáticamente, y un largo etc.

En este contexto, podemos pensar en una comunicación ciudad-ciudadano mucho más sencilla, en el sentido de no buscar ningún fin más allá del propio intercambio de conocimiento. Imaginemos una ciudad donde cada edificio, cada parque, cada árbol, tiene algo que contar a los ciudadanos que deseen conocerlo. Qué son, qué hacen ahí, quién los puso, qué historias han generado, etc. Al mismo tiempo, los ciudadanos pueden querer aportar su conocimiento a la ciudad, aportando su granito de arena a la historia que cada elemento puede contar. Estaríamos ante una ciudad que, gracias al uso de las nuevas tecnologías, estaría dotada de una dimensión totalmente nueva a disposición de sus habitantes y de aquellos visitantes que quieran conocerla.

Y las posibilidades crecen si pensamos en nuevas tecnologías emergentes como la realidad aumentada. Pensar en poder ver a través de tus dispositivos una proyección de cómo era antes un edificio, de cómo puede ser un árbol cuando crezca, de qué batalla sucedió en estas murallas, etc. nos hace pensar en una ciudad no sólo inteligente, sino al mismo tiempo dotada de conocimiento colectivo al servicio de la gente. Esto generaría, además, un grado de consciencia mayor de los ciudadanos respecto a su entorno, impulsando la conservación de su patrimonio natural, cultural y artístico.

Es en este modelo de ciudad donde encaja nuestro proyecto. Un proyecto de comunicación ciudad-ciudadano que pretende servir de modelo para que una ciudad como Pamplona aproveche su potencial como ciudad inteligente.

## 1.4 DISPOSITIVOS MÓVILES

No cabe duda de que nos encontramos en la era de internet. Hoy en día una gran parte del planeta dispone de acceso a la red y, desde hace ya algunos años, son los pequeños aparatos que todos llevamos en el bolsillo los que nos proveen de esa puerta de acceso. El disponer de dispositivos móviles con conexión a internet y con la capacidad de almacenar en su interior un gran número de aplicaciones que hacen uso de ella ha abierto un abanico de posibilidades para que los desarrolladores pongan todo tipo de herramientas a disposición de los usuarios: desde aplicaciones que permiten comunicarnos con gente de todo el mundo de manera inmediata hasta juegos online, pasando por todo tipo de utilidades como navegadores, mapas, brújulas, etc. que aprovechan las posibilidades de estos dispositivos y las ponen al servicio de los usuarios.

Si bien son sobre todo las nuevas generaciones las que más utilizan estos dispositivos, observamos como poco a poco los más mayores también se van acercando a ellos y van dejando atrás los miedos y desconfianzas para entregarse a los beneficios que pueden ofrecerles. Realizar pues un proyecto en el ámbito de los dispositivos móviles supone llegar a todo tipo de usuarios, y esto es algo que se ha tenido en cuenta a la hora de implementar las diferentes soluciones adoptadas.

Hay que tener en cuenta, además, que hay mucha variedad en cuanto a dispositivos. Hoy en día nos encontramos con teléfonos y tabletas de todo tipo que no sólo se diferencian en tamaño, peso, tipo de cámara, etc. sino que incorporan además distintos sistemas operativos, por lo que una de las primeras tareas a la hora de realizar cualquier tipo de proyecto orientado a estos dispositivos será decidir qué espectro de dispositivos se va a querer cubrir y cómo.

Actualmente los sistemas operativos más extendidos son Android de Google e iOS de Apple, que copan el 96,3% de los dispositivos. A continuación vemos una tabla con los datos de mercado de los principales sistemas operativos móviles en los dos últimos años.

| Sistema operativo | Volumen de unidades 2014 | Cuota de mercado 2014 | Volumen de unidades 2013 | Cuota de mercado 2013 | Cambio año a año |
|---|---|---|---|---|---|
| **Android** | 1.059,3 | 81,5% | 802,2 | 78,7% | 32,0% |
| **iOS** | 192,7 | 14,8% | 153,4 | 15,1% | 25,6% |
| **Windows Phone** | 34,9 | 2,7% | 33,5 | 3,3% | 4,2% |
| **BlackBerry** | 5,8 | 0,4% | 19,2 | 1,9% | -69,8% |
| **Otras** | 7,7 | 0,6% | 2,3 | 0,2% | 234,8% |
| **Total** | 1.300,4 | 100% | 1.018,7 | 100% | 27,7% |

**Tabla 1.1 Distribución de los S.O. móviles en 2013 y 2014**

Como vemos en la tabla, hay un sistema operativo que destaca por encima del resto: Android. El 81,5% de los dispositivos vendidos en 2014 incorporan este sistema operativo, por lo que podemos decir que es sin duda la plataforma más extendida y por lo tanto la más interesante de cara a realizar una aplicación que pueda usar el máximo número de usuarios.

Por otra parte, prácticamente la totalidad de los dispositivos disponen de una cámara fotográfica incorporada, así como otro tipo de sensores, como pueden ser: acelerómetro, giroscopio, magnetómetro, etc. Una de las utilidades más potentes y que más se utilizan es el GPS, sistema de posicionamiento global, para lo que la mayoría de dispositivos incorporan un chip específico. Este chip permite la geolocalización del dispositivo y, con ello, el uso de esa localización para mostrar al usuario información personalizada como puede ser el tiempo que hace en el lugar donde se encuentra, los restaurantes que tiene a su alrededor, etc.

Al realizar el proyecto, se han tenido en cuenta todo este tipo de aspectos, tratando de llegar al máximo número de dispositivos y aprovechar las tecnologías que la mayoría de ellos incorporan. Para ello, y como se detallará más adelante, se ha optado por desarrollar una aplicación para el sistema operativo Android, ya que de este modo habrá un mayor número de usuarios que puedan hacer uso de ella.

Al mismo tiempo, y para que no sólo aquellos visitantes del parque que dispongan de un móvil Android puedan disfrutar del recorrido interactivo, se ha implementado un sitio web, de manera que desde cualquier dispositivo conectado a internet y con un navegador web instalado pueda leer toda la información desde su dispositivo, así como ver las imágenes y escuchar los audios.

## 1.5 ESTADO DEL ARTE

A la hora de realizar un proyecto de esta naturaleza, conviene echar un vistazo a las alternativas que se ofrecen actualmente en este ámbito. En este caso, se ha enfocado la investigación a las soluciones disponibles en la zona de Pamplona, ya que es ésta la ciudad en la que vamos a desarrollar nuestra actuación.

- **Recorridos por la ciudad:**

    - **El paseo de las murallas**:

      Se trata de un recorrido de unos cinco kilómetros de longitud, que ofrece un paseo bordeando las murallas de Pamplona a lo largo del cual se va mostrando información acerca de los bastiones, baluartes, portales, medias lunas, revellines, fuertes…

      Este paseo comienza en el fortín de San Bartolomé, donde podemos encontrar el cartel que vemos en la figura 1.3. En él se muestra información acerca del fortín, en varios idiomas: castellano, euskera, francés e inglés. También se puede observar un mapa de Navarra con indicaciones de dónde se encuentran otro tipo de fortificaciones similares. Por último, podemos encontrar una serie de códigos QR, uno para cada idioma, que enlazan con la web http://www.murallasdepamplona.com/.

En el resto de zonas de interés del recorrido encontramos unas mesas cartel con la información en los cuatro idiomas y también en braille.


**Figura 1.3 Cartel indicador del paseo de las murallas**

- **Aplicaciones móviles:**

    - **Pamplona Me Gusta**:


**Figura 1.4 Aplicación Pamplona Me Gusta**

Se trata de la aplicación oficial de turismo del Ayuntamiento de Pamplona. Se puede descargar de manera gratuita y recoge toda la oferta turística de la ciudad para poder planificar un viaje y conocerla. Esta aplicación se enmarca en el trabajo que el consistorio está realizando hacia el objetivo de acercarse a la ciudad inteligente, aplicando las nuevas tecnologías en la gestión municipal.

La aplicación Pamplona Me Gusta trata de informar sobre los recursos turísticos, culturales y patrimoniales de la ciudad y facilitar así al visitante un plan de viaje en caso de que esté pensando en viajar a Pamplona. La aplicación es gratuita y está disponible tanto en Google Play para dispositivos Android, como en App Store para iOS.

Como características, podemos destacar:

- Funcionamiento offline: se puede navegar a través de ella sin problema aunque no tengamos conexión a internet.
- Geoposicionamiento: los datos pueden ser mostrados en función de la localización del visitante, facilitando su orientación.
- Disponible en cuatro idiomas: castellano, euskera, francés e inglés.

Desde la página principal, que podemos observar en la figura 1.4, el usuario puede acceder a las siguientes secciones:

- “Qué tengo cerca”: muestra de un vistazo, a través de un plano, la ubicación de los hitos principales de la ciudad en lo que a monumentos, edificios y lugares históricos y turísticos se refiere. La ubicación va a acompañada del nombre del lugar y de una breve descripción de su importancia.
- “Imprescindibles”: incluye información sobre lugares significativos de la ciudad como las plazas Consistorial y del Castillo, la catedral, la calle Estafeta, la Ciudadela, la Taconera o el Caballo Blanco, ineludibles para quien quiera conocer los encantos de Pamplona. El Ayuntamiento, a través de esta aplicación, ofrece sugerencias para los visitantes.
- "Qué hacer en 1, 2 o 3 días”: es una propuesta pensada para que los visitantes no se pierdan lo principal de la ciudad, en función de cuánto tiempo permanezcan en ella. Murallas, gastronomía, recorrido del encierro o paseos por los parques son algunas de las recomendaciones que se dan para disfrutar plenamente de la ciudad. Asimismo, Pamplona cuenta con una serie de paseos, podríamos decir temáticos, que repasan distintos aspectos de la historia o la cultura de la ciudad.
- “Paseos por la ciudad”: se describen rutas por la Ciudadela y las murallas, por el paseo fluvial del Arga, por el Camino de Santiago, por la ruta del encierro o por la llamada Pamplona monumental y artística, con los principales hitos de cada una de ellas.

- **Zumaiako audio gida**:


**Figura 1.5 Aplicación Zumaiako Audio Gida**

En lo que a aplicaciones con audio-guías se refiere, no hemos encontrado ningún ejemplo en Pamplona, pero sí en la localidad guipuzcoana de Zumaia. Mediante esta aplicación, podemos escuchar audios explicativos acerca de varios puntos de interés de la costa de Zumaia, en tres idiomas: euskera, castellano e inglés.

Los recorridos, compuestos cada uno de una serie de audios, con sus respectivas imágenes y nombres, deben ser descargados previamente, como podemos observar en la figura 1.5, y no saltan automáticamente al llegar al lugar, sino que tenemos que ser nosotros los que activamente reproduzcamos el audio correspondiente.

# 2 OBJETIVOS

## 2.1 OBJETIVOS GENERALES

Este PFC pretende ofrecer, como ya se ha dicho anteriormente, un recorrido interactivo a través del parque de Trinitarios de Pamplona. Desde Opera Ingeniería se nos planteó la idea de seleccionar una serie de árboles del parque que ellos considerasen representativos y almacenar cierta información de cada uno de ellos. Esta información es la que después se deberá mostrar al usuario, bien a través de la aplicación Android, como de la web. Como complemento a estas dos opciones, y sirviendo de punto de referencia para cada uno de los árboles, se colocarán unas placas informativas al lado de cada árbol, que contendrán información de cada uno de los árboles, tanto en euskera como en castellano, y un código QR que servirá de punto de entrada, tanto para la aplicación como para la web.

La idea del recorrido es que se pueda hacer a tres niveles diferentes:

- **Recorrer el parque simplemente leyendo las placas informativas:**
  Esta forma de recorrer el parque está pensada para aquellos usuarios que no dispongan de un dispositivo móvil con el que leer los códigos QR ni hacer uso de la aplicación. Mediante los textos incluidos en las propias placas, se podrá realizar el recorrido sin tener que utilizar ningún tipo de dispositivo. El objetivo es no restringir la que creemos que es una experiencia muy interesante a aquellos usuarios más habituados al uso de las nuevas tecnologías, y que otro tipo de perfil, como pueden ser el de las personas mayores que a menudo visitan el parque, también disfrute del recorrido.

- **Recorrer el parque leyendo los códigos QR:**
  El segundo nivel incorporaría ya un cierto nivel de interacción, ya que resultará necesario disponer de un dispositivo móvil y de una aplicación que disponga de un lector de códigos QR. Esta forma de recorrer el parque hará una pequeña distinción entre aquellos usuarios que lean los códigos desde una aplicación cualquiera, y aquellos que lo hagan mediante el lector incorporado en nuestra aplicación. Los primeros serán redirigidos a la url que contiene el código, por lo que se les mostrará la página web de nuestro sitio que corresponda con esa placa. Los segundos, en cambio, no saldrán de la aplicación, ya que ésta detectará a qué elemento corresponde la url del código QR y simplemente mostrará la pantalla dentro de la aplicación que corresponde a ese elemento.

- **Recorrer el parque mediante la audio-guía:**
  Esta forma de recorrer el parque debe ser el modo más “automático” de hacerlo, ya que el objetivo es que los visitantes no interactúen activamente con el parque, sino que sea éste, por medio de nuestra aplicación, quien interactúe con el visitante, contándole las cosas conforme vaya visitando los distintos puntos establecidos. Estos puntos coincidirán con las localizaciones exactas de las placas de cada árbol, intentando hacer que cuando el visitante se encuentre lo más cerca posible del árbol, salte el audio correspondiente.

Para la correcta implementación de este recorrido, será necesaria la realización de tres tareas, cada una de las cuales tendrá sus objetivos particulares: el servidor y la base de datos; el sitio web; y la aplicación Android.

## 2.2 SERVIDOR Y BASE DE DATOS

Esta parte del proyecto tendrá como objetivo la creación de la infraestructura necesaria para almacenar y servir toda la información que va a ser mostrada tanto en el sitio web como en la aplicación Android. Además de la propia información de cada árbol, en nuestro servidor almacenaremos también las imágenes y audios que se muestran tanto en el sitio web como en la aplicación.

Ambos clientes (web y app) deberán ser capaces de comunicarse con el servidor y éste deberá servirles la información solicitada de una manera adecuada.

La información de los árboles, los audios, y las páginas que componen el sitio web deberán estar tanto en euskera como en castellano, por lo que deberemos crear tanto tablas duplicadas para la base de datos, como diferentes directorios para los audios y las páginas web.

La información y los textos referentes a cada árbol serán recopilados por Opera Ingeniería a partir de bibliografía que se referencia en el apartado 8.

Deberemos recopilar las imágenes que vayamos a almacenar, las cuales serán descargadas de internet respetando escrupulosamente las licencias y copyright correspondientes.

También deberemos grabar los audios para cada árbol, lo cual haremos mediante una grabadora de voz e intentando conseguir la colaboración de gente dispuesta a prestarnos su voz.

## 2.3 APLICACIÓN ANDROID

En esta parte, se pretende crear una aplicación bilingüe para Android que permita no sólo visualizar la información, sino permitir que el visitante pueda realizar el recorrido interactivo de la forma en la que hemos apuntado anteriormente: mediante los códigos QR y la audio-guía.

Esta aplicación debe proveer la información del recorrido de una manera clara e intuitiva. Deberá mostrar los elementos tanto en una lista como en un mapa del parque, localizados mediante marcadores.

El lector QR será implementado haciendo uso de una librería externa, ya que no entra dentro de los objetivos del PFC la implementación de un detector de códigos QR.

La audio-guía deberá hacer uso de los sistemas de geolocalización presentes en el dispositivo para ofrecer un seguimiento lo más preciso posible que posibilite una grata experiencia al usuario.

Procuraremos cuidar al máximo el diseño de la aplicación, teniendo en cuenta la temática y los posibles usuarios de la misma.

## 2.4 SITIO WEB

Como complemento a la aplicación deberemos realizar un sitio web alojado en nuestro hosting, al igual que el servidor. Este sitio web tiene que tener como objetivos: por una parte, servir de alternativa para aquellos usuarios con dispositivos no compatibles con la aplicación, pero que sí dispongan de un lector QR. Por otro lado, el objetivo es también que el sitio web sirva para dar a conocer el parque, su recorrido interactivo y la aplicación. Así pues, además de mostrar información de los elementos, deberemos informar de qué es lo que se ofrece con este recorrido, de qué formas se puede realizar, y tendrá que aparecer una explicación de la aplicación y un enlace para descargarla.

# 3 ANÁLISIS

Como ya se ha comentado anteriormente, se propone realizar: por una parte, una aplicación para dispositivos Android que facilite el recorrer el parque de una forma interactiva mediante un lector QR y una audio-guía; y por otra parte, realizar un sitio web que muestre el contenido y sirva de alternativa para la gente que no disponga de la aplicación.

Los requisitos que se enumeran a continuación surgen en su mayoría en las primeras reuniones entre las partes que hemos colaborado en este proyecto, una vez se consigue tener una idea clara de qué es lo que se quiere conseguir.

Al componerse el proyecto de tres partes claramente diferenciadas, se han dividido los distintos requisitos en partes: una para los requisitos del servidor, otra para la aplicación Android y la última para el sitio web.

## 3.1 REQUISITOS FUNCIONALES

En este apartado se describen todas las funcionalidades básicas que deben cumplir las dos partes que componen este proyecto. Estos requisitos definen el comportamiento del sistema y de todos sus componentes.

- **Servidor:**
  - La información de cada árbol deberá estar almacenada en una base de datos alojada en el servidor, tanto en euskera como en castellano.
  - El servidor deberá permitir realizar consultas a la base de datos y devolver los resultados de la consulta a aquellos clientes que se lo pidan.
  - El servidor deberá alojar las imágenes que vayan a ser utilizadas por los diferentes clientes.
  - El servidor deberá alojar los audios que vayan a ser utilizados por los diferentes clientes.

- **Aplicación Android:**
  - La aplicación deberá ser capaz de acceder a los datos de cada árbol almacenados en la base de datos externa. Para ello tendrá que disponer de un sistema de conexión con nuestro servidor que permita la obtención de estos datos.
  - La aplicación deberá poder mostrar un listado de todos los árboles que componen el recorrido, por orden alfabético.
  - La aplicación también mostrará un mapa del parque indicando la localización geográfica de cada árbol por medio de un marcador.
  - El usuario deberá poder seleccionar uno de los árboles tanto desde la lista como desde el mapa y ver la información asociada a ese árbol.
  - La aplicación debe permitir escanear códigos QR e identificar el árbol al que hacen referencia.

- La aplicación dispondrá de una audio-guía que rastreará de forma continua la localización geográfica del dispositivo, y lanzará un audio cuando detecte que el dispositivo está lo suficientemente cerca del árbol correspondiente.
- Toda la aplicación deberá ser bilingüe, en castellano y en euskera.
- La aplicación deberá estar disponible de manera gratuita para su descarga en Play Store.

- **Sitio web:**
  - Las páginas web que componen el sitio deberán ser capaces de acceder a los datos almacenados en la base de datos.
  - La página principal deberá mostrar información general acerca del parque y del recorrido interactivo.
  - Desde la página principal deberá ser posible descargar la aplicación Android, mediante un enlace a su página de Google Play.
  - Desde la página principal se debe mostrar un listado de todos los árboles que componen el recorrido, por orden alfabético.
  - Cada una de las páginas de los distintos árboles deberán mostrar toda la información asociada a ese árbol.

## 3.2 REQUISITOS NO FUNCIONALES

Los requisitos que se enumeran a continuación, aun no siendo esenciales para el funcionamiento de la aplicación, ayudarán a conseguir que el rendimiento de la misma sea el mejor posible. Además, estos requisitos también definen cómo deberían ser las interfaces de usuario para que la experiencia del mismo sea también la mejor posible.

- **Aplicación Android:**
  - La aplicación se deberá visualizar correctamente en todo tipo de dispositivos compatibles, adaptando sus interfaces a los diferentes tamaños de pantalla para ofrecer una mejor experiencia de usuario.
  - La navegación a través de las diferentes secciones de la aplicación deberá ser lo más intuitiva posible, facilitando la transición entre ellas y proveyendo de menús e instrucciones claras acerca de las opciones que cada sección ofrece.
  - El diseño de la aplicación deberá ser claro, sencillo y con una cohesión lo más alta posible, para que de esta manera el usuario se sienta más cómodo y sea más agradable su utilización.
  - La aplicación deberá ofrecer la máxima fluidez posible, lo cual implica que las operaciones que conllevan conexión con el servidor y que puedan demorarse en el tiempo, se deban realizar siempre en segundo plano, para no interrumpir así el hilo principal de la aplicación, y el usuario no note que la aplicación se “congela” en ningún momento.
  - La aplicación deberá hacer una gestión adecuada de la memoria disponible que ofrece el dispositivo, lo cual implica crear una caché para las imágenes de un tamaño proporcional a la memoria disponible. De esta manera se evitarán desbordamientos de memoria, que son un problema muy a tener en cuenta a

la hora de realizar una aplicación para dispositivos móviles, ya que existen grandes diferencias entre ellos.

- Debido a que el uso del GPS en la audio-guía se debe realizar de manera continua, se deberá tener muy en cuenta el uso de la batería que ello conlleva y llegar a un equilibrio entre precisión y bajo consumo, que permita ofrecer una localización suficientemente precisa minimizando el consumo.

- **Sitio web:**
  - El sitio web deberá visualizarse correctamente desde cualquier dispositivo, desde ordenadores hasta móviles, adaptándose a cualquier tamaño de pantalla. Para ello se deberá adoptar un diseño adaptable.
  - El diseño del sitio web deberá ser, al igual que el de la aplicación, lo más limpio, claro y coherente posible, haciendo de la navegación por el mismo, una experiencia fácil, cómoda y agradable.

## 3.3 REQUISITOS HARDWARE

Los requisitos hardware para cada una de las partes que componen el proyecto son:

- **Servidor:**
  - El servidor deberá tener espacio suficiente para almacenar todos los ficheros que lo componen, así como el sitio web, imágenes y audios.
  - No se han establecido requisitos previos en cuanto a peticiones por unidad de tiempo al servidor, aunque se presupone que el servidor debe ser capaz de dar soporte a un número adecuado de peticiones.

- **Aplicación Android:**
  - El dispositivo que haga uso de la aplicación debe disponer de conexión a Internet a la hora de utilizar la aplicación. Se ha decidido que sea así, para tener que evitar el introducir todas las imágenes y los audios en la propia aplicación, lo que supondría un aumento excesivo en el tamaño de la misma. Además, la conexión a internet resulta indispensable tanto para mostrar los mapas de Google Maps como para registrar las peticiones de geolocalización de la audio-guía.
  - Para hacer uso del lector de códigos QR, el dispositivo deberá contar al menos con una cámara fotográfica incorporada.
  - Para hacer uso de la audio-guía, el dispositivo debe disponer de un chip receptor GPS incorporado.

- **Sitio web:**
  - El sitio web debe ser visible con independencia del dispositivo desde el que se haga.

## 3.4 REQUISITOS SOFTWARE

Los requisitos software para cada una de las partes que componen el proyecto son:

- **Aplicación Android:**
  - Para hacer uso de la aplicación hará falta disponer de un teléfono con sistema operativo Android en su versión 4.0.3 o superior. Elegir esta versión como requisito mínimo para poder usar la aplicación nos permite, como se puede ver en la siguiente tabla, llegar a un 94,1% de los dispositivos y al mismo tiempo, ofrecer algunas características y hacer uso de algunas librerías externas que exigen esa versión mínima para poder ser implementadas.

| Version | Nombre | API | Distribución |
|---|---|---|---|
| 2.2 | Froyo | 8 | 0.3% |
| 2.3.3 - 2.3.7 | Gingerbread | 10 | 5.6% |
| 4.0.3 - 4.0.4 | Ice Cream Sandwich | 15 | 5.1% |
| 4.1.x | Jelly Bean | 16 | 14.7% |
| 4.2.x | | 17 | 17.5% |
| 4.3 | | 18 | 5.2% |
| 4.4 | KitKat | 19 | 39.2% |
| 5.0 | Lollipop | 21 | 11.6% |
| 5.1 | | 22 | 0.8% |

**Tabla 3.1 Distribución de las versiones de Android a 1 de junio de 2015**

# 4 DISEÑO

En este apartado se muestra cómo se han diseñado los distintos componentes del proyecto en base a los requisitos especificados en la etapa de análisis.

## 4.1 BASE DE DATOS

Nuestro modelo de base de datos es muy simple: tan sólo se necesita modelar una entidad, que representará cada uno de los árboles del recorrido. Como necesitamos almacenar información tanto en castellano como en euskera, hemos creado dos tablas equivalentes, una para cada idioma.

En la siguiente figura vemos la estructura de la tabla element_eus. La estructura es la misma para element_spa, tan solo cambia el contenido de los campos. Además, la clave primaria en ambos casos es la misma, el campo id, que tendrá el mismo valor para cada elemento en ambas tablas, pudiendo así identificar un elemento por su id y buscarlo en cualquiera de las tablas obteniendo el mismo resultado.

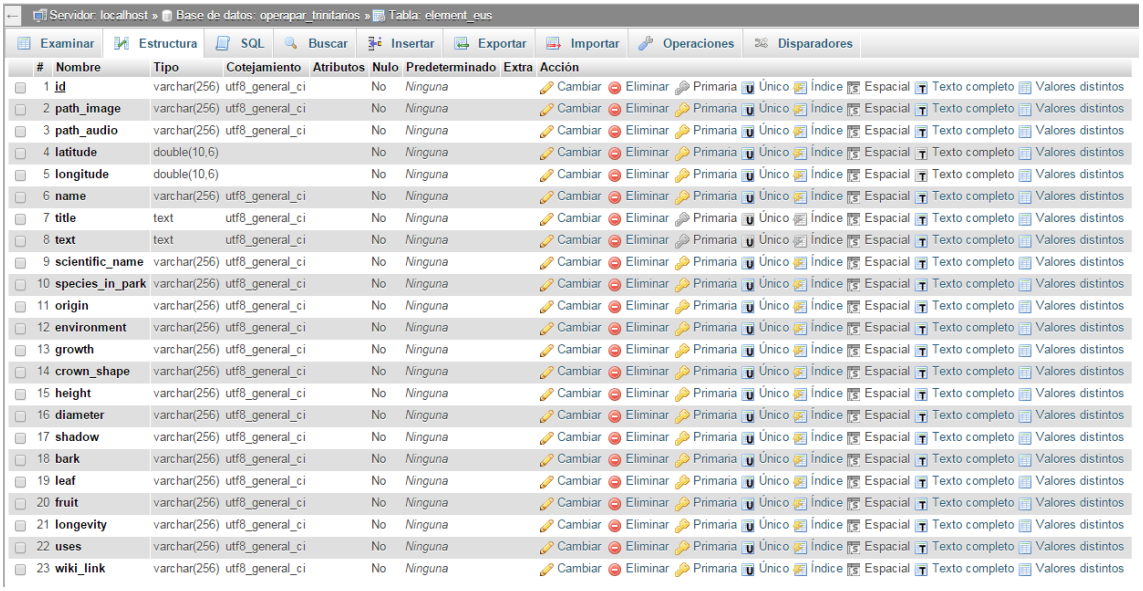



**Figura 4.1 Estructura de la tabla element_eus**

Los campos de la tabla se corresponden con toda la información que necesitamos almacenar de cada elemento. La mayoría de los campos se corresponden con propiedades de cada árbol como la altura, el diámetro, etc., pero también almacenamos otra información de cada elemento que conviene destacar:

- **path_image**: especifica la ruta desde el directorio raíz de nuestro servidor hasta la imagen del elemento.
- **path_audio**: especifica la ruta desde el directorio raíz de nuestro servidor hasta el audio del elemento.
- **latitude**: especifica la latitud exacta a la que se encuentra la placa informativa de este elemento.

- **longitude**: especifica la longitud exacta a la que se encuentra la placa informativa de este elemento.

## 4.2 CONEXIÓN CLIENTE-SERVIDOR

La conexión entre nuestro servidor y los dos clientes (aplicación y web) sigue el siguiente esquema:

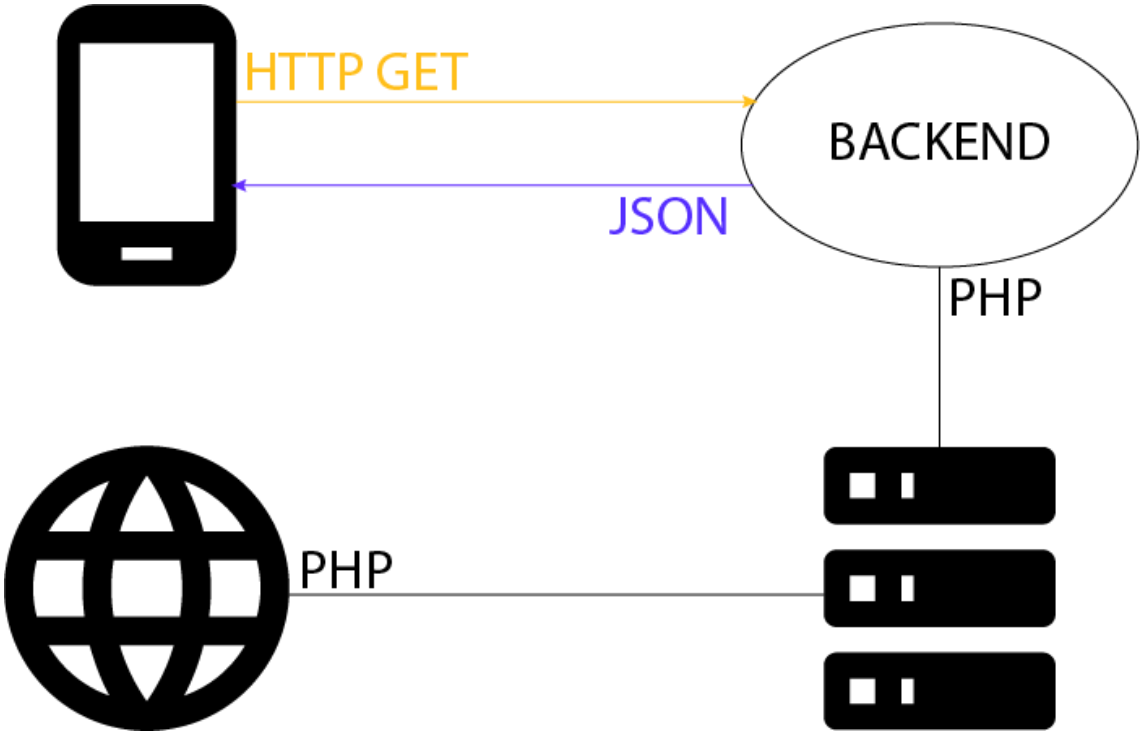



**Figura 4.2 Esquema de conexión cliente-servidor**

Como podemos ver, la conexión con el servidor se realiza de distinta forma para la aplicación y para el sitio web. Mientras que este último se compone de ficheros php que están alojados en el propio servidor y que son capaces de acceder a los datos y mostrarlos al navegador convirtiéndolos en las etiquetas HTML correspondientes, para la aplicación móvil hace falta disponer de una serie de funciones intermediarias que puedan recibir peticiones de la aplicación y devolver los datos en un formato adecuado para su tratamiento por parte del cliente.

En nuestro caso, para esa parte intermediaria, se han creado una serie de ficheros PHP que recibirán peticiónes HTTP de tipo GET y devolverán los resultados en formato JSON.

JSON (JavaScript Object Notation) es un formato ligero para el intercambio de datos que resulta muy apropiado para este tipo de situaciones debido a su simplicidad y a las ventajas que esto supone.

# 5 IMPLEMENTACIÓN

En este apartado se describe cómo se han implementado las soluciones adoptadas, tratando de cumplir con los requisitos definidos en el apartado 3 y siguiendo el diseño descrito en el apartado 4. Para ello, se ha dividido el apartado en subsecciones que describen la implementación de cada una de las partes que componen el proyecto.

## 5.1 SERVIDOR Y BASE DE DATOS

Para la implementación del servidor y la base de datos ha sido necesaria la contratación de un hosting donde alojar todos los elementos que lo conforman: base de datos, ficheros php, imágenes, audios y sitio web (compuesto a su vez de varios ficheros, carpetas e imágenes).

Durante los primeros meses, se contrató un hosting gratuito en 2freeHosting. Esta empresa ofrece un servicio a priori bastante bueno para resultar gratuito. A decir verdad, este hosting nos sirvió perfectamente durante la etapa de desarrollo, pero tenía aspectos clave que no nos permitían continuar con él de cara a la implementación final:

- La velocidad de descarga de las imágenes no era el deseable para un proyecto en el que las imágenes tienen tanto poder visual como éste.
- Sufría constantes caídas.
- El soporte técnico no era el más adecuado.

Por todo esto, y de cara a la implementación y pruebas finales, se ha optado por contratar un hosting compartido de pago en Raiola Networks, una empresa de hosting española que tiene muy buenas críticas y que nos ofrece:

- 10 GB de almacenamiento
- 80 GB de transferencia
- 3 DB MySQL
- 512 MB de RAM
- 50% de 1 CPU

El precio del mismo asciende a 4,95 €/mes e incluye el dominio que se desee gratis durante el primer año. Para la realización del proyecto se ha decidido registrar el dominio operaparques.es y asociarlo a nuestro servicio de hosting. El coste ha corrido a cargo de la empresa Opera Ingeniería.

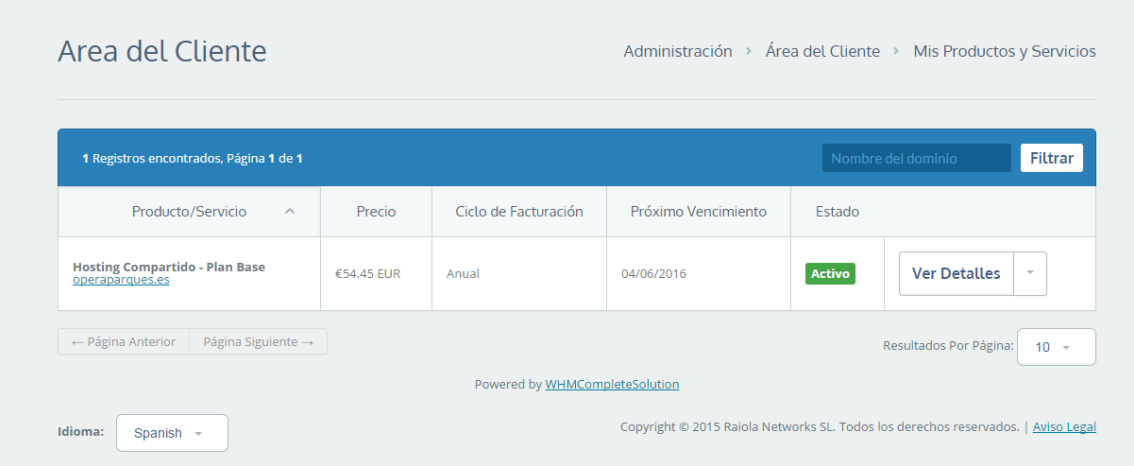



**Figura 5.1 Área del Cliente de Raiola Networks**

En este hosting, hemos creado una base de datos para almacenar los datos de los árboles. Como hemos visto en el anterior apartado, nuestro modelo tan solo necesita de una tabla, aunque debido a que toda la información debe estar disponible tanto en castellano como en euskera, se han creado finalmente dos tablas idénticas pero con los valores en sus respectivos idiomas. Así pues, tenemos:

- Base de datos operapar_trinitarios.
  - Tabla element_eus: contiene la información de los árboles en euskera.
  - Tabla element_spa: contiene la información de los árboles en castellano.

Para la creación de la base de datos, de las tablas, y el poblado de las mismas, se ha utilizado la herramienta phpMyAdmin.

Además de la creación de la base de datos, el servidor contiene los siguientes elementos:

- Fichero **DBHandler.php**:

```php
<?php
function connectDB() {
    $server = ini_get("mysql.default_host");
    $user = "operapar_julen";
    $pass = "[redacted]";
    $db = "operapar_trinitarios";

    $conn = mysqli_connect ( $server, $user, $pass, $db ) or die ( 'Connection to the DB failed.' );
    return $conn;
}
function disconnectDB($conn) {
    $close = mysqli_close ( $conn ) or die ( 'Disconnection from the DB failed.' );
    return $close;
}
function getObjectSQL($sql) {
    $conn = connectDB ();
    mysqli_set_charset ( $conn , "utf8" );
    $result = mysqli_query ( $conn, $sql ) or die ( 'Error in query: ' . $query );
    while ( $r = mysqli_fetch_assoc ( $result ) ) {
        $rows [] = $r;
    }
    disconnectDB ( $conn );
    return $rows;
}

?>
```

**Figura 5.3 Fichero DBHandler.php**

Este fichero, ubicado en la carpeta libs, contiene las funciones necesarias para conectarse a nuestra base de datos, y además contiene una función llamada getObjectSQL que recibe una petición SQL como parámetro y devuelve el resultado de la consulta a la base de datos.

Este fichero es utilizado tanto por el backend de la aplicación Android, como por aquellas páginas del sitio web que necesiten hacer consultas a la base de datos.

- Carpeta **images**: carpeta que contiene todas las imágenes que vamos a utilizar tanto en la aplicación como en el sitio web.

Estas imágenes han sido obtenidas en la red social flickr y, como se indica en la bibliografía, están bajo licencia Creative Commons y pueden ser utilizadas para fines no comerciales.

- Carpeta **audio_spa**: carpeta que contiene todos los archivos de audio en castellano que vamos a utilizar tanto en la aplicación como en el sitio web.

- Carpeta **audio_eus**: carpeta que contiene todos los archivos de audio en euskera que vamos a utilizar tanto en la aplicación como en el sitio web.

Estos archivos de audio han sido generados utilizando una grabadora de voz. Para su grabación, se ha contado con la colaboración de una serie de personas que han prestado su voz al proyecto.

- **Backend de la aplicación**: Comprende todos los ficheros ubicados en la carpeta android_backend, que son los siguientes:

  - **get_all_elements_spa.php**

```php
<?php

header("Cache-Control: public, max-age=3600");

include $_SERVER ['DOCUMENT_ROOT'] . "/trinitarios/libs/DBHandler.php";

$sql = "SELECT * FROM element_spa ORDER BY name";
$myObj = getObjectSQL ( $sql );
echo json_encode ( $myObj );

?>
```

**Figura 5.4 Fichero get_all_elements_spa.php**

Este fichero, como vemos, incluye las funciones de DBHandler para poder enviar una consulta a la base de datos. En este caso, la consulta consiste en obtener todos los elementos de la tabla element_spa. Como vemos, la respuesta se envía en formato JSON al cliente.

  - **get_all_elements_eus.php**: equivalente al anterior pero para la tabla element_eus.

  - **get_element_spa.php**

```php
<?php

header("Cache-Control: public, max-age=3600");

include $_SERVER ['DOCUMENT_ROOT'] . "/trinitarios/libs/DBHandler.php";

if (isset ( $_GET ["id"] )) {

    $id= "'" . $_GET ["id"] . "'";
    $sql = "SELECT * FROM element_spa WHERE id = $id";
    $myObj = getObjectSQL ( $sql );
    echo json_encode ( $myObj );
}

?>
```

**Figura 5.5 Fichero get_element_spa.php**

Este fichero, al contrario que los anteriores, recibe un parámetro GET, que utiliza para realizar una consulta en la base de datos, buscando en la tabla element_spa aquel elemento con id igual al parámetro recibido. La respuesta es enviada en formato JSON.

- **get_element_eus.php**: equivalente al anterior pero para la tabla element_eus.

Como se ha podido observar, todos estos ficheros contienen un header HTTP de control de caché para que el cliente almacene las respuestas durante una hora.

## 5.2 APLICACIÓN ANDROID

### 5.2.1 ENTORNO DE DESARROLLO

Para el desarrollo de la aplicación se ha utilizado el IDE Android Studio, un IDE pensado expresamente para la plataforma Android y que está basado en IntelliJ IDEA de JetBrains. Está desarrollado por Google y la última versión estable es la 1.2.1.1

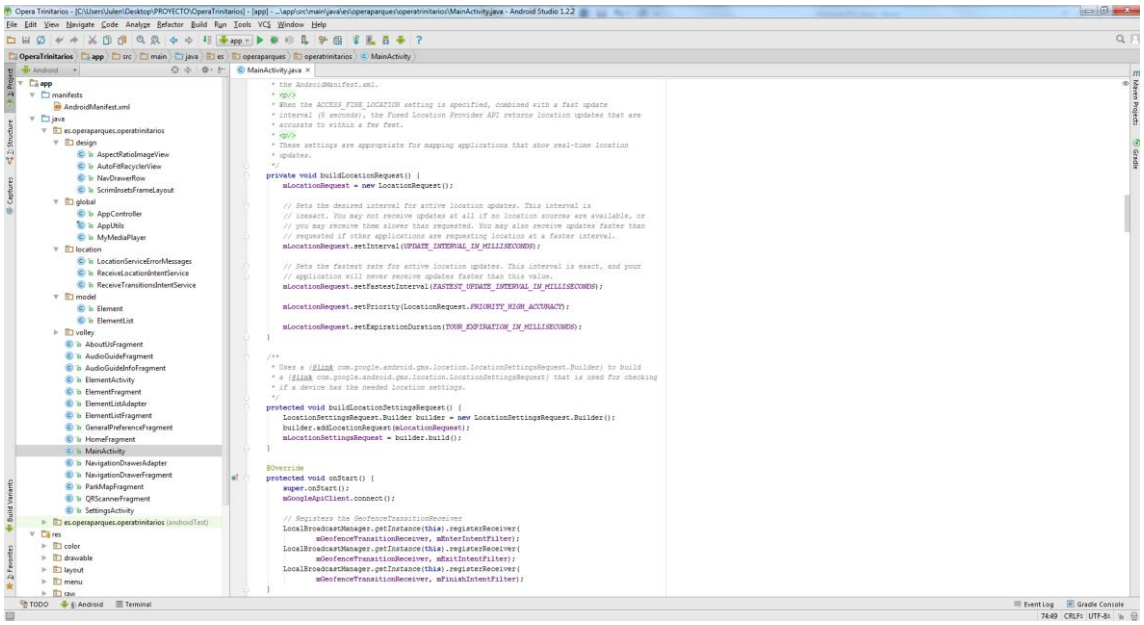


**Figura 5.6 IDE Android Studio**

La aplicación se ha desarrollado utilizando la versión 22 del SDK de Android y mediante el lenguaje de programación JAVA.

Tanto los nombres de las clases y ficheros, como los de los métodos e incluso los comentarios han sido escritos en inglés.

### 5.2.2 ESTRUCTURA DE LA APLICACIÓN

Para el desarrollo de una aplicación Android, una de las primeras decisiones a tomar es pensar en cómo se va a estructurar la misma. En Android, existen una serie de componentes básicos mediante los cuales se van construyendo y estructurando las aplicaciones.

Entre ellas, se encuentran las Activities y los Fragments. Las Activities son componentes que proveen una pantalla determinada mediante la cual el usuario puede interactuar con la aplicación. Así pues, una aplicación Android normalmente se compone de un puñado de Activities que se comunican y están de alguna manera relacionadas entre sí. Al mismo tiempo, cada Activity puede alojar en su interior uno o más Fragments, que como su propio nombre indica, representan un fragmento de la interfaz de usuario o del comportamiento de toda la Activity que lo contiene.

Otros componentes pueden ser los Services, cuya función es llevar a cabo tareas en segundo plano.

En este caso, las Activities y los Fragments que las componen son:

- **MainActivity**: la Activity principal de la aplicación. Es la primera pantalla que se le muestra al usuario y su función es la de controlar la navegación entre las diferentes secciones de la aplicación. Para facilitar esta navegación, esta Activity contiene un cajón de navegación lateral que funciona a modo de menú. Esta activity contiene los siguientes Fragments:

    - **NavigationDrawerFragment**: este Fragment se encarga de mostrar el cajón de navegación lateral de la aplicación. Contiene un NavigationDrawerAdapter que se encarga de rellenar la lista de elementos del cajón lateral y de captar los eventos de click sobre ellos.

    - **HomeFragment**: este Fragment se encarga de mostrar la sección de Inicio, controlando todo el funcionamiento de la misma.
    - **ParkMapFragment**: este Fragment se encarga de mostrar la sección de Mapa, controlando todo el funcionamiento de la misma.
    - **ElementListFragment**: este Fragment se encarga de mostrar la sección de Árboles, controlando todo el funcionamiento de la misma.
    - **QRScannerFragment**: este Fragment se encarga de mostrar la sección de Escáner QR, controlando todo el funcionamiento de la misma.
    - **AudioGuideFragment**: este Fragment se encarga de mostrar la sección de Audio guía, controlando todo el funcionamiento de la misma.
    - **AboutUsFragment**: este Fragment se encarga de mostrar la sección de Acerca de, controlando todo el funcionamiento de la misma.

- **ElementActivity:** esta Activity se encarga de mostrar la información de un elemento. A esta Activity se llega desde distintos puntos de MainActivity y tiene a ésta como Activity padre, por lo que al navegar hacia atrás desde esta Activity se vuelve a MainActivity.

- **SettingsActivity:** esta Activity se encarga de mostrar los ajustes de la aplicación, que en este caso se limitan a la selección del idioma.

### 5.2.3 VOLLEY

Para realizar la comunicación con el servidor, se ha hecho uso de la librería externa Volley. Esta librería, desarrollada por los propios desarrolladores de Google, permite una comunicación HTTP más rápida y sencilla. Entre otras cosas, Volley ofrece:

- Planificación automática de peticiones.
- Múltiples conexiones concurrentes.
- Cache automática tanto en memoria como en disco

Para hacer uso de ella, se ha copiado y compilado el fichero volley.jar.

Las llamadas se realizan de la siguiente manera.

```java
/**
 * Creates and delivers a request to the server in order to
 * all the elements.
 */
public static void volleyRequestAll(Context context, final volleyRequestListener caller) {
    String url;

    SharedPreferences sharedPref = PreferenceManager.getDefaultSharedPreferences(context);
    String language = sharedPref.getString(SettingsActivity.KEY_PREF_LANGUAGE, "");

    if (language != null) {
        switch (language) {
            case CODE_LANGUAGE_SPA:
                url = URL_GET_ALL_ELEMENTS_SPA;
                break;
            case CODE_LANGUAGE_EUS:
                url = URL_GET_ALL_ELEMENTS_EUS;
                break;
            default:
                //url = URL_GET_ALL_ELEMENTS_ENG;
                url = URL_GET_ALL_ELEMENTS_SPA;
                break;
        }
    } else {
        //url = URL_GET_ALL_ELEMENTS_ENG;
        url = URL_GET_ALL_ELEMENTS_SPA;
    }

    final CustomArrayRequest jsArrRequest = new CustomArrayRequest(url, null,
            new Response.Listener<JSONArray>() {

                @Override
                public void onResponse(JSONArray response) {
                    Log.d(TAG, "Response: " + response);
                    // Send the response to the caller.
                    caller.onResponse(response);
                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError response) {
            Log.d(TAG, "Error: " + response.toString());
            // Send the error to the caller.
            caller.onError(response);
        }
    });
    AppController.getInstance().addToRequestQueue(jsArrRequest);
}
```

**Figura 5.7 Método volleyRequestAll en AppUtils**

Este método realiza una petición solicitando la lista de todos los elementos. Lo primero que hacemos es obtener el idioma actual de la aplicación para saber la url del fichero del backend al que tenemos que dirigirnos. Después realiza la petición y se queda a la espera de la respuesta. La respuesta llega de forma asíncrona, no interrumpiendo así el hilo principal de la aplicación. Cuando llega la respuesta, se la enviamos a aquel componente que haya llamado al método.

### 5.2.4 LECTOR QR

Para el lector QR hemos utilizado la librería ZBar, que provee de las clases necesarias para detectar este tipo de códigos. Al detectar un código QR, lo que hacemos es lo siguiente.

```
@Override
public void handleResult(Result result) {

    String url = result.getContents();
    String elementId = "";
    if ((url.toLowerCase().contains(AppUtils.URL_SERVER_DOMAIN_EUS.toLowerCase())) &&
            (url.length()>AppUtils.URL_SERVER_DOMAIN_EUS.length())){
        elementId = url.substring(AppUtils.URL_SERVER_DOMAIN_EUS.length(), url.length()-4);
    } else if ((url.toLowerCase().contains(AppUtils.URL_SERVER_DOMAIN.toLowerCase())) &&
            (url.length()>AppUtils.URL_SERVER_DOMAIN.length())){
        elementId = url.substring(AppUtils.URL_SERVER_DOMAIN.length(), url.length()-4);
    }

    // Start a new activity to show the element in detail.
    ((MainActivity) getActivity()).requestElement(elementId);
}
```

**Figura 5.8 Método handleResult en QRScannerFragment**

Cuando el lector detecta un código QR, se llama a este método, pasándole como parámetro el contenido del código QR. Como en el código lo que hay es una url, ya sea la de castellano o la de euskera, y sabemos que en ambos casos el final de la url coincide con el id del elemento más .php lo que hacemos es extraer esa última parte, quitarle los últimos 4 caracteres y así poder pedirle a nuestra Activity que nos muestre el elemento correspondiente a ese id.

### 5.2.5 AUDIO-GUÍA

En cuanto a la implementación de la audio-guía, conviene describir brevemente cómo funciona el servicio de geolocalización que utilizamos para saber si el usuario se encuentra cerca de determinado punto, y lanzar así el audio correspondiente.

Para lograrlo hemos hecho uso de las llamadas Geofences, o geo-vallas, que básicamente son círculos en torno a un punto geográfico que sirven para determinar si una persona se encuentra a menos de cierta distancia de un sitio.

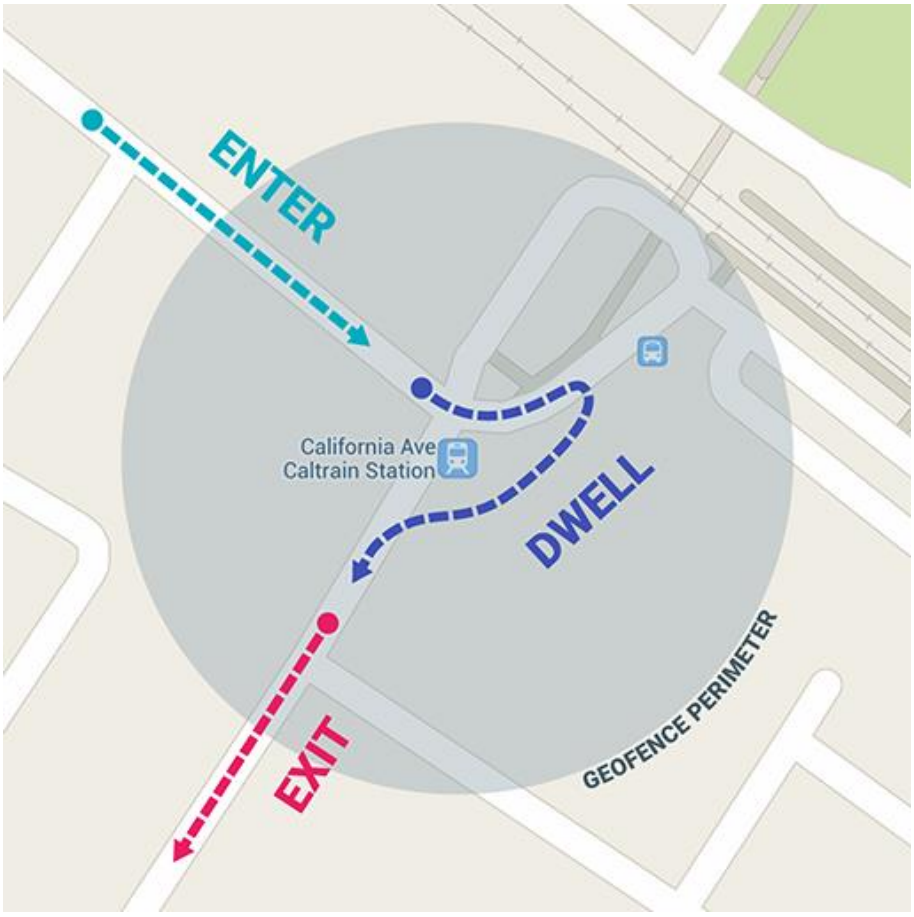



**Figura 5.9 Geofence**

Las Geofences forman parte de los servicios de localización de Android, y para utilizarlas tenemos que registrarlas haciendo uso del método addGeofences de la clase GeofencingApi. Una vez registrada, podremos recibir mensajes a través de un servicio en segundo plano cada vez que los servicios de localización detecten que el usuario entra o sale de la zona. Para definir una geo-valla basta con establecer la longitud y latitud del centro de la zona y el radio deseado. Además, hay que destacar que para poder registrar este tipo de geo-vallas, el dispositivo tiene que tener activa la opción de alta precisión en las opciones de ubicación.

Para este proyecto, se tuvo que decidir qué radio era el adecuado con el fin de garantizar un correcto funcionamiento en la mayoría de los dispositivos. El hacerlo muy pequeño puede favorecer a aquellos móviles que tengan más precisión, ya que sólo escucharán el audio al estar realmente cerca del lugar, pero perjudicaría a los que no tienen tanta precisión en su detector GPS, ya que las geo-vallas no registran una transición de entrada o de salida si el grado de confianza de que así sea no supera un determinado valor. El hacer el radio muy grande, en cambio, haría que casi todos los móviles pudiesen recibir estas transiciones, pero a costa de que las zonas fuesen muy poco precisas e incluso se solaparan unas con otras.

Buscando un equilibrio y con base a distintas pruebas realizadas con diferentes móviles, se estableció un radio de 25m para las zonas. De esta manera el recorrido y las zonas para las geo-vallas queda como vemos en la siguiente figura.

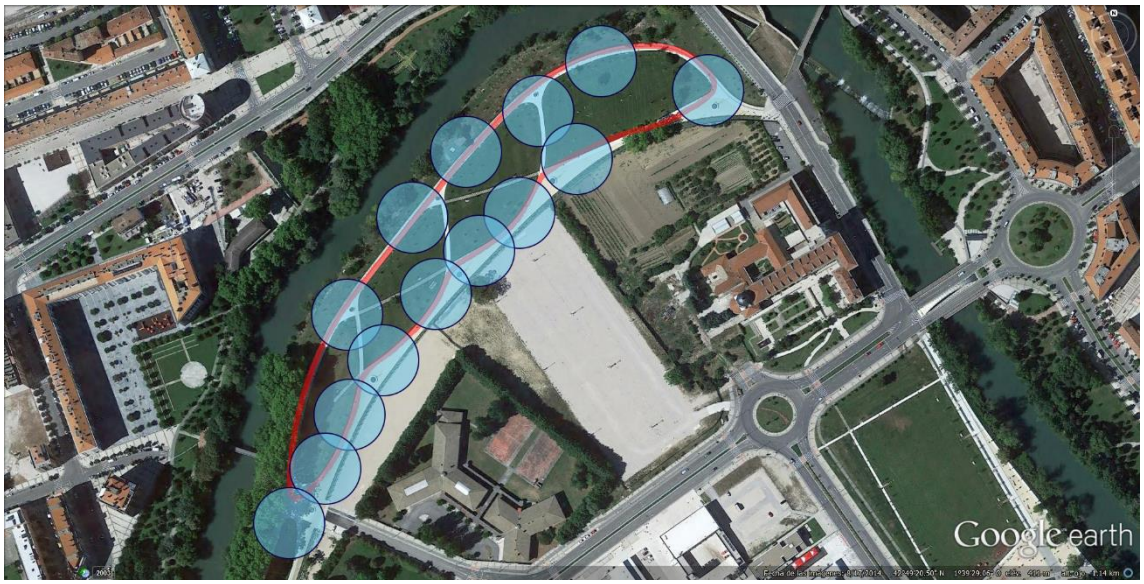



**Figura 5.10 Geofences en el parque**

Como vemos, el grado de solape es muy bajo, por lo que no hay riesgo de que salte el audio que no corresponde, y las zonas son lo suficientemente amplias para que la mayoría de los dispositivos consigan funcionar correctamente.

Además de registrar las Geofences, para la audio-guía necesitamos solicitar la localización del dispositivo de forma continua y en segundo plano, para lo que necesitaremos hacer uso del método requestLocationUpdates de la clase FusedLocationApi.

Al terminar la audio-guía, tanto las geo-vallas como las peticiones de localización son detenidas y eliminadas con los respectivos métodos removeGeofences y removeLocationUpdates.

Todas estas llamadas necesitan de un objeto de tipo GoogleApiClient que tenemos que crear y conectar antes de hacer cualquiera de esas llamadas.

A continuación vamos a ver cómo se han implementado estas llamadas desde nuestra MainActivity.

```
@Override
public void onAudioGuideStarted(ElementList elementList) {
    mElementList = elementList;
    mRequest = REQUEST_START;
    if (!servicesConnected()) {
        return;
    }
    if (!mGoogleApiClient.isConnected()) {
        Toast.makeText(this, "GoogleApiClient is not connected yet. Try again", Toast.LENGTH_SHORT).show();
        return;
    }
    checkLocationSettings();
}
```

**Figura 5.11 Método onAudioGuideStarted en MainActivity**

Cuando el usuario solicita empezar la audio-guía, lo primero que hacemos es ver si estamos conectados a los Google Services mediante el método servicesConnected. Después comprobamos que nuestro objeto GoogleApiClient está conectado. Por último, llamamos al método checkLocationSettings, que lo que hace es comprobar que tenemos activado el modo de alta precisión y, si no lo tenemos, nos pregunta si lo queremos activar.

Más tarde, si todo está correcto, llamamos al método startGeofencing.

```
private void startGeofencing() {
    // Get the geofences used. Geofence data is hard coded in this sample.
    populateGeofenceList();

    addGeofences();
}
```

**Figura 5.12 Método startGeofencing en MainActivity**

```
/**
 * This sample hard codes geofence data. A real app might dynamically create geofences based on
 * the user's location.
 */
public void populateGeofenceList() {

    for (Element element : mElementList.getList()) {

        Log.d(AppUtils.TAG, "Building geofence of: " + element.getName());

        mGeofenceList.add(new Geofence.Builder()
                // Set the request ID of the geofence. This is a string to identify this
                // geofence.
                .setRequestId(element.getId())

                        // Set the circular region of this geofence.
                .setCircularRegion(element.getLatitude(), element.getLongitude(),
                        GEOFENCE_RADIUS_IN_METERS)

                        // Set the expiration duration of the geofence. This geofence gets automatically
                        // removed after this period of time.
                .setExpirationDuration(TOUR_EXPIRATION_IN_MILLISECONDS)

                        // Set the transition types of interest. Alerts are only generated for these
                        // transition. We track entry and exit transitions in this sample.
                .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER |
                        Geofence.GEOFENCE_TRANSITION_EXIT)

                        // Create the geofence.
                .build());
    }
}
```

**Figura 5.13 Método populateGeofenceList en MainActivity**

El método populateGeofenceList es el encargado de crear una lista de geo-vallas y añadir una geo-valla por cada elemento de nuestra lista de elementos. Como vemos, el centro lo define la longitud y la latitud de nuestro elemento, el radio es una constante que como hemos dicho tiene valor 25 y en cuanto al tiempo de expiración, lo hemos establecido en 2 horas.

```
private void addGeofences() {

    try {
        LocationServices.GeofencingApi.addGeofences(
                mGoogleApiClient,
                // The GeofenceRequest object.
                getGeofencingRequest(),
                // A pending intent that that is reused when calling removeGeofences(). This
                // pending intent is used to generate an intent when a matched geofence
                // transition is observed.
                AppController.getInstance().getGeofencePendingIntent()
        ).setResultCallback((status) -> {
                if (status.isSuccess()) {
                    startLocationUpdates();
                    Log.i(AppUtils.TAG, "Geofence adding succeeded");
                } else {
                    showErrorDialog();
                    Log.e(AppUtils.TAG, "Error adding geofences");
                }
        }); // Result processed in onResult().
    } catch (SecurityException securityException) {
        // Catch exception generated if the app does not use ACCESS_FINE_LOCATION permission.
        logSecurityException(securityException);
    }
}
```

**Figura 5.14 Método addGeofences en MainActivity**

El método addGeofences es el que registra todas las geo-vallas utilizando el método addGeofences. El PendingIntent que obtenemos con el método getGeofencePendingIntent() es el que nos permite recibir en el servicio ReceiveTransitionsIntentService los mensajes de entrada y salida de las distintas zonas. Esto lo vemos en la siguiente figura.

```
/**
 * Gets a PendingIntent to send with the request to add or remove Geofences. Location Services
 * issues the Intent inside this PendingIntent whenever a geofence transition occurs for the
 * current list of geofences.
 *
 * @return A PendingIntent for the IntentService that handles geofence transitions.
 */
public PendingIntent getGeofencePendingIntent() {

    // Reuse the PendingIntent if we already have it.
    if (mGeofencePendingIntent != null) {

        // Return the existing intent
        return mGeofencePendingIntent;
    } else {

        // Create an Intent pointing to the IntentService
        Intent intent = new Intent(this, ReceiveTransitionsIntentService.class);

        // We use FLAG_UPDATE_CURRENT so that we get the same pending intent back when calling
        // addGeofences() and removeGeofences().
        mGeofencePendingIntent = PendingIntent.getService(this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);
        return mGeofencePendingIntent;
    }
}
```

**Figura 5.15 Método getGeofencePendingIntent en AppController**

Después de registrar las geo-vallas, y de forma análoga, registramos las peticiones de actualizaciones de localización.

```
private void startLocationUpdates() {

    // Request location updates.
    LocationServices.FusedLocationApi.requestLocationUpdates(
            mGoogleApiClient, mLocationRequest,
            AppController.getInstance().getLocationPendingIntent())
            .setResultCallback(new ResultCallback<Status>() {
                @Override
                public void onResult(Status status) {
                    if (status.isSuccess()) {
                        mTourStarted = true;
                        SharedPreferences.Editor editor = mSharedPreferences.edit();
                        editor.putBoolean(AppUtils.KEY_TOUR_STARTED, mTourStarted);
                        editor.apply();
                        if (mAudioGuideFragment != null && mAudioGuideFragment.isAdded()) {
                            mAudioGuideFragment.audioGuideStarted();
                        }
                        Log.i(AppUtils.TAG, "Location request succeeded");
                    } else {
                        Log.e(AppUtils.TAG, "Error requesting location updates");
                    }
                }
            });
}
```

**Figura 5.16 Método startLocationUpdates en MainActivity**

Finalmente, pasamos a explicar qué sucede cuando el servicio ReceiveTransitionsIntentService recibe un mensaje de transición.

```
// Get the type of transition (entry or exit)
int transition = event.getGeofenceTransition();

// Test that a valid transition was reported
if (transition == Geofence.GEOFENCE_TRANSITION_ENTER) {

    List<Geofence> geofences = event.getTriggeringGeofences();
    String elementId = geofences.get(0).getRequestId();

    Boolean alreadyVisited =
            sharedPreferences.getBoolean(AppUtils.KEY_VISITED + elementId, false);

    if (!alreadyVisited) {

        SharedPreferences.Editor editor = sharedPreferences.edit();
        int elementsLeft = sharedPreferences.getInt(AppUtils.KEY_ELEMENTS_LEFT, 0);
        elementsLeft--;
        editor.putInt(AppUtils.KEY_ELEMENTS_LEFT, elementsLeft);
        editor.putBoolean(AppUtils.KEY_VISITED + elementId, true);
        editor.putBoolean(AppUtils.KEY_TOUR_UPDATED, true);
        editor.apply();

        // Get the element.
        AppUtils.volleyRequest(getApplicationContext(), elementId, this);
    }

} else if (transition == Geofence.GEOFENCE_TRANSITION_EXIT) {
    // Broadcasts the Intent to receivers in this app.
    Intent localIntent = new Intent(AppUtils.BROADCAST_ACTION_EXIT);
    LocalBroadcastManager.getInstance(this).sendBroadcast(localIntent);
} else {
    // Always log as an error
    Log.e(AppUtils.TAG,
            "Geofence transition error. Invalid type {transition}");
}
```

**Figura 5.17 Método onHandleIntent en ReceiveTransitionsIntentService**

Como vemos, lo primero que hacemos es comprobar que se trata de una transición de entrada en una zona, que es lo que realmente nos interesa. Más tarde, obtenemos el id del elemento al que corresponde la geo-valla, y que previamente le habíamos asociado al registrarla. Comprobamos si no se había visitado ya durante esta audio-guía y en caso negativo procedemos a actualizar los valores de las variables que utilizamos para controlar el estado de la audio-guía. Por último, solicitamos al servidor el elemento que corresponde a ese id. Necesitamos el elemento para obtener su nombre, el cual mostraremos en la notificación que lanzaremos, y para obtener la ruta del audio, que también lanzaremos desde el propio servicio.

```
@Override
public void onResponse(JSONArray response) {
    try {
        Element element = new Element(response.getJSONObject(0));
        // Log the transition type and a message
        Log.d(AppUtils.TAG,
                "Near: {element.getName()}");
        Log.d(AppUtils.TAG,
                "Click to see in detail");

        // Send notification.
        sendNotification(element);
        // Launch audio.
        launchAudio(element);
        // Broadcasts the Intent to receivers in this app.
        Intent localIntent = new Intent(AppUtils.BROADCAST_ACTION_ENTER);
        LocalBroadcastManager.getInstance(this).sendBroadcast(localIntent);
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

**Figura 5.18 Método onResponse en ReceiveTransitionsIntentService**

Cuando recibimos la respuesta del servidor, construimos el elemento y lanzamos la notificación y el audio. Por último, lanzamos un broadcast a la aplicación para que el Fragment de la audio-guía, en caso de estar visible en ese momento, actualice su interfaz para mostrar el nuevo elemento como visitado. En caso de no estar visible, al volver a serlo, el Fragment comprobará si debe actualizarse en base a una de las variables que utilizamos para controlar el estado de la audio-guía.

Una vez que la audio-guía finaliza, bien porque se han visitado todos los elementos o bien porque el usuario le da al botón de finalizar, el borrado y detención tanto de las geo-vallas como de las actualizaciones se produce de una manera similar al registro, pero cambiando los métodos a llamar.

### 5.2.6 CLASES

Además de las clases propias de la estructura básica de la aplicación, como las Activity y las Fragment que ya se han definido en el apartado 5.2.1, la aplicación hace uso de otras clases, las cuales están organizadas por paquetes y que se enumeran a continuación.

- **Paquete design:** Aquí se encuentran aquellas clases creadas con la intención de crear componentes personalizados para la interfaz de la aplicación.

    - **AspectRatioImageView:** Un ImageView es un componente de la interfaz que se encarga de dibujar una imagen. Esta clase es un ImageView que, al dibujarse, calcula la altura que debe tener la imagen en función de la anchura que tiene y de la relación de aspecto deseada. Esto nos permite asegurarnos de que todas las imágenes de la aplicación tengan una relación de aspecto 16:9.

    - **AutoFitRecyclerView:** Un RecyclerView es un componente de la interfaz que se encarga de dibujar una lista de elementos en una ventana. Esta clase es un RecyclerView que adapta su Grid (forma en la que se colocan los elementos) al tamaño de la ventana. Así conseguimos que en pantallas grandes los elementos se muestren en más de una columna en vez de en una sola.

    - **NavDrawerRow:** Esta clase representa una fila en el cajón de navegación. Nos sirve para tener una referencia del nombre, icono y Fragment asociado a cada fila.

    - **ScrimInsetsFrameLayout:** Clase necesaria para permitir que el cajón de navegación se dibuje incluso por encima de la barra de notificaciones del dispositivo. Su función es puramente estética.

- **Paquete global:** Aquí se encuentran aquellas clases que son utilizadas por el resto de la aplicación.

    - **AppController:** Clase singleton que hereda de la clase Application, por lo que su ciclo de vida no está atado a ninguna Activity sino a toda la aplicación. Nos sirve para almacenar aquellos objetos que necesitamos que sean permanentes a lo largo de toda la aplicación, como algunos objetos usados por la librería Volley y por algunos otros usados por la audio-guía. Además, contiene el método utilizado para establecer el idioma de la aplicación y que mostramos a continuación:

```
/**
 * This method sets the current locale of the app. It gets the language to be applied from
 * the SharedPreferences.
 */
public void setLocale() {
    Locale locale;

    SharedPreferences sharedPref = PreferenceManager.getDefaultSharedPreferences(this);
    String language = sharedPref.getString(SettingsActivity.KEY_PREF_LANGUAGE, "");

    if (language != null) {
        switch (language) {
            case AppUtils.CODE_LANGUAGE_SPA:
                locale = new Locale("es", "ES");
                break;
            case AppUtils.CODE_LANGUAGE_EUS:
                locale = new Locale("eu", "ES");
                break;
            default:
                locale = new Locale("en", "US");
                break;
        }
    } else {
        locale = new Locale("en", "US");
    }

    Locale.setDefault(locale);
    Configuration config = new Configuration();
    config.locale = locale;
    getBaseContext().getResources().updateConfiguration(config, null);
}
```

**Figura 5.7 Método setLocale**

- **AppUtils:** esta clase contiene todas las constantes y métodos estáticos que se utilizan en toda la aplicación. También contiene los métodos utilizados para hacer peticiones al servidor mediante la librería Volley.

- **MyMediaPlayer:** esta clase singleton nos permite asegurarnos de que nunca se crea más de una instancia de la clase MediaPlayer, encargada de reproducir sonidos. Esto nos sirve para que todos los audios que se lancen en nuestra aplicación puedan ser capaces de detener un audio que esté sonando en ese momento y así no solaparse entre ellos.

- **Paquete location:** Aquí se encuentran aquellas clases utilizadas para geolocalizar el móvil y detectar los puntos durante la audio-guía. El funcionamiento de ésta se ha explicado en el apartado 5.2.6.

    - **LocationServiceErrorMessages:** Contiene referencias a strings asociados a errores del servicio de localización de Google.

    - **ReceiveLocationIntentService:** Un servicio que recibe en segundo plano las actualizaciones de localización de los Servicios de Google, durante la audio-guía. No hace nada con la localización recibida, pero este servicio es necesario para que las actualizaciones se reciban incluso estando el móvil con la pantalla apagada.

    - **ReceiveLocationIntentService:** Un servicio que recibe en segundo plano mensajes si entramos o salimos de una de las geo-vallas definidas para la audio-guía.

- **Paquete model:** contiene las clases que representan nuestro modelo de datos.

    - **Element:** Representa un elemento. Contiene los mismos campos que la tabla element de la base de datos, y su constructor, como vemos en la siguiente figura, toma como parámetro un objeto JSONObject. De esta manera, a partir de una petición al servidor, recibimos el elemento en formato JSON y con él creamos un objeto de esta clase para poder mostrarlo después al usuario.

```
public Element(JSONObject o) {
    try {
        id = o.getString("id");
        pathImage = AppUtils.URL_SERVER_DOMAIN + o.getString("path_image");
        pathAudio = AppUtils.URL_SERVER_DOMAIN + o.getString("path_audio");
        latitude = o.getDouble("latitude");
        longitude = o.getDouble("longitude");
        name = o.getString("name");
        title = o.getString("title");
        text = o.getString("text");
        scientificName = o.getString("scientific_name");
        speciesInPark = o.getString("species_in_park");
        origin = o.getString("origin");
        environment = o.getString("environment");
        growth = o.getString("growth");
        crownShape = o.getString("crown_shape");
        height = o.getString("height");
        diameter = o.getString("diameter");
        shadow = o.getString("shadow");
        bark = o.getString("bark");
        leaf = o.getString("leaf");
        fruit = o.getString("fruit");
        longevity = o.getString("longevity");
        uses = o.getString("uses");
        wikiLink = o.getString("wiki_link");
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

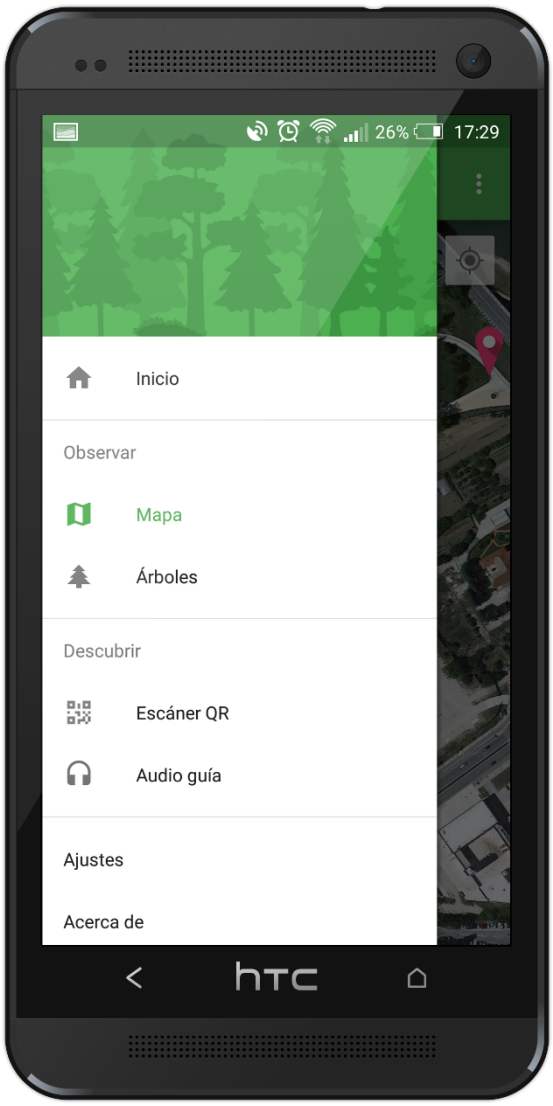
**Figura 5.8 Constructor de Element**

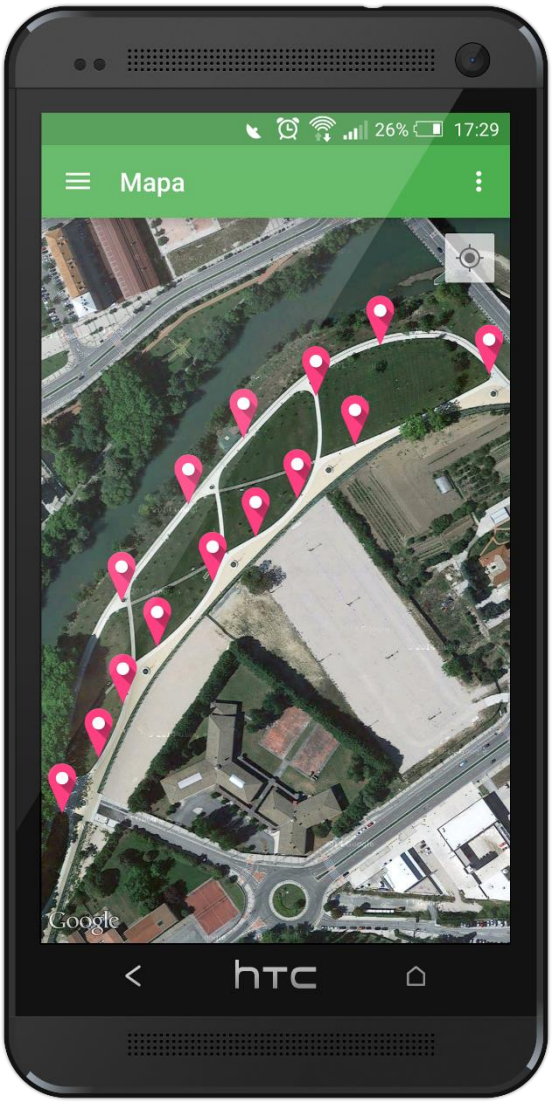
- **ElementList:** Representa una lista de elementos. Contiene tanto una List de Element como un Map<String, Element> que permitirá obtener un Element a partir de su id. Se construye a partir de un JSONArray, como vemos en la siguiente figura.

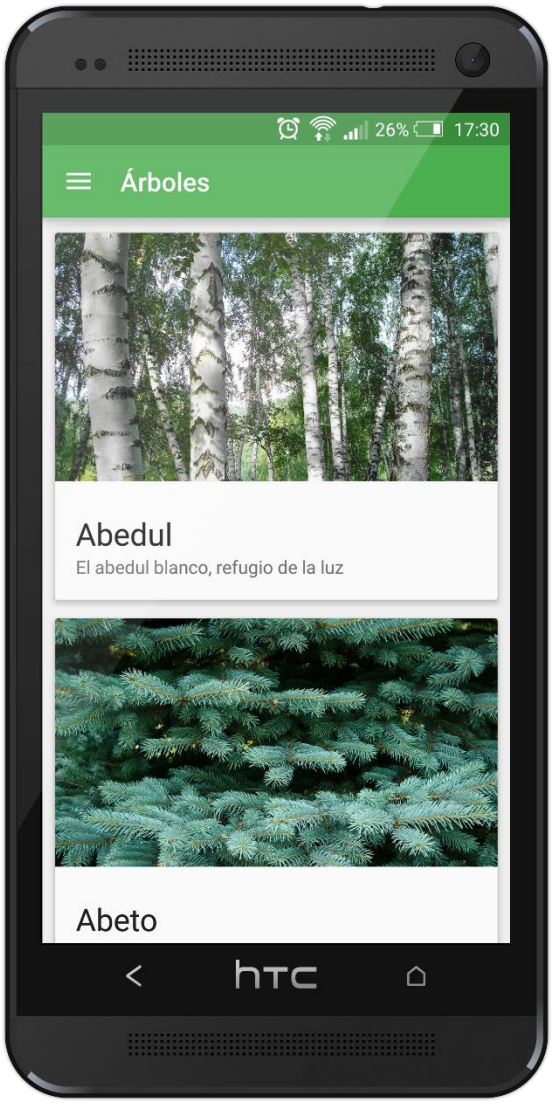
```
/**
 * A list of elements.
 */
private List<Element> mElementList;

/**
 * A map of id-element pairs.
 */
private Map<String, Element> mElementMap;

public ElementList(JSONArray jsArray) {
    mElementList = new ArrayList<>();
    mElementMap = new HashMap<>();
    for (int i = 0; i < jsArray.length(); i++) {
        try {
            addElement(new Element(jsArray.getJSONObject(i)));
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}

public void addElement(Element element) {
    mElementList.add(element);
    mElementMap.put(element.getId(), element);
}
```

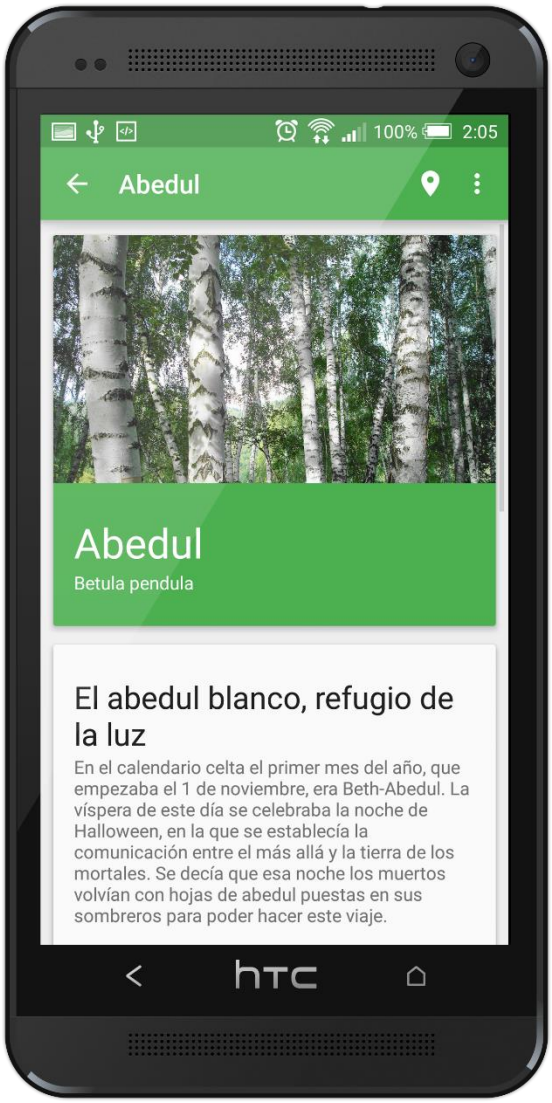
**Figura 5.9 Constructor de ElementList**

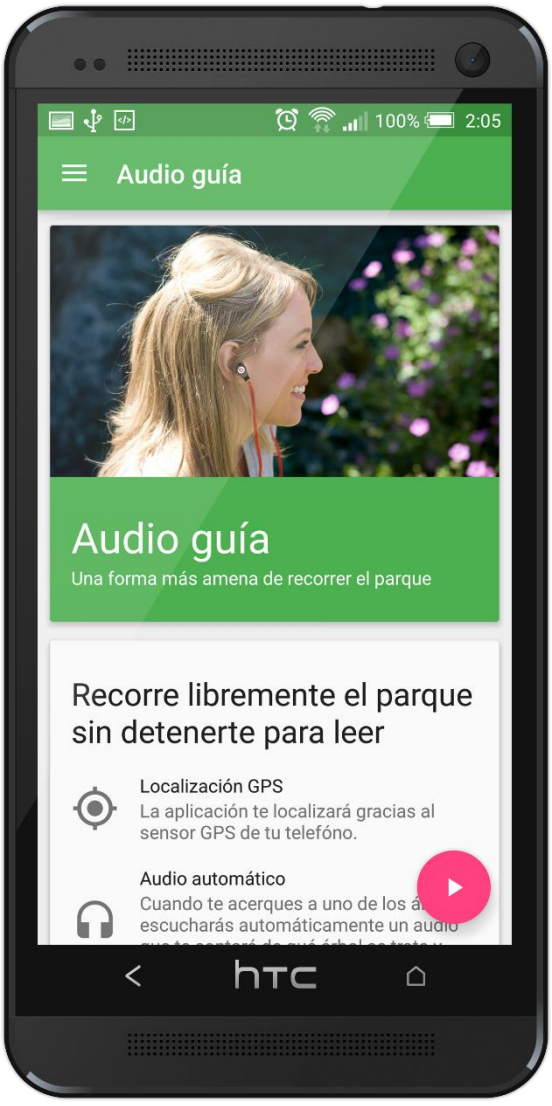
### 5.2.7 PANTALLAS DE LA APLICACIÓN

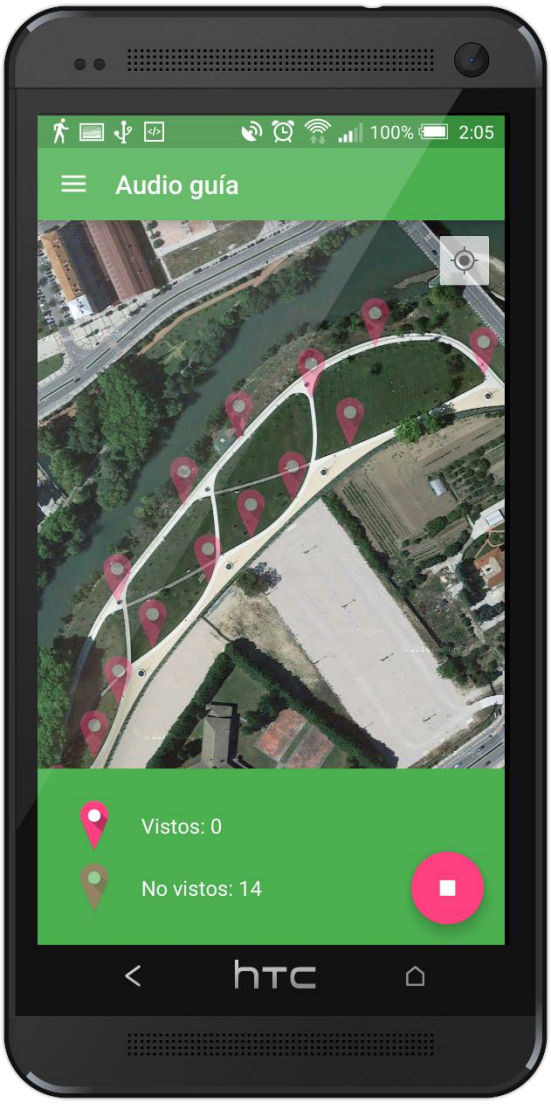


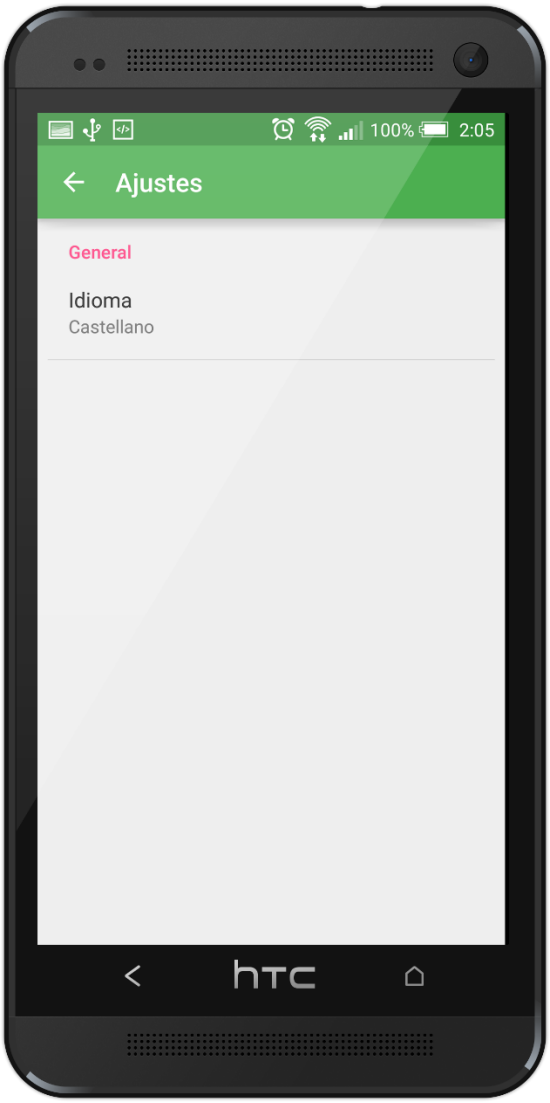



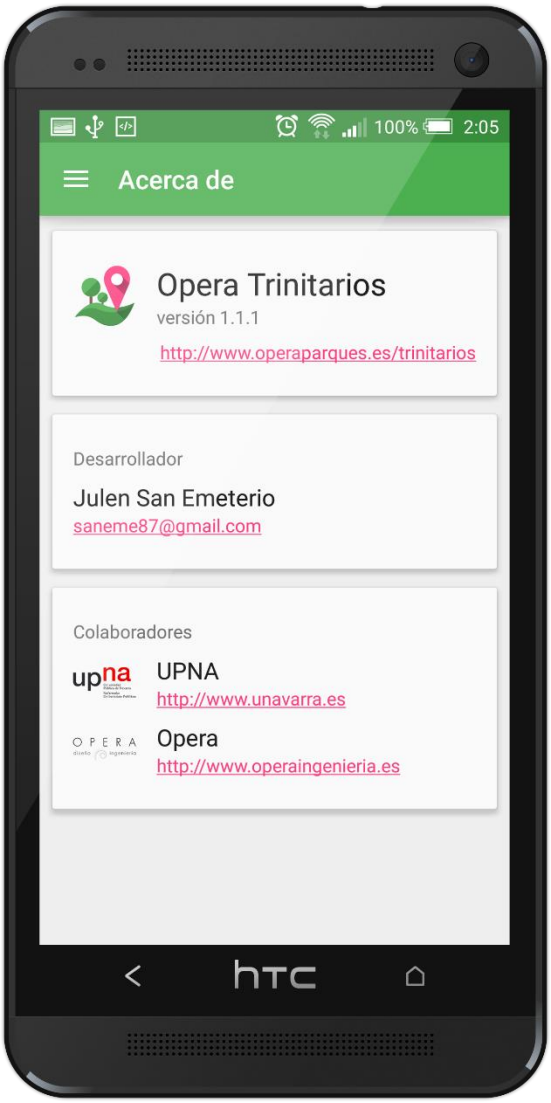




100% 2:05
Audio guía
Audio guía
Una forma más amena de recorrer el parque
Recorre libremente el parque sin detenerte para leer
Localización GPS
La aplicación te localizará gracias al sensor GPS de tu telefóno.
Audio automático
htc


100% 2:05
Audio guía
Vistos: 0
No vistos: 14
htc


100% 2:05
Ajustes
General
Idioma
Castellano
htc


100% 2:05
Acerca de
Opera Trinitarios
versión 1.1.1
http://www.operaparques.es/trinitarios
Desarrollador
Julen San Emeterio
saneme87@gmail.com
Colaboradores
UPNA
http://www.unavarra.es
Opera
http://www.operaingenieria.es
htc

## 5.3 SITIO WEB

El sitio web ha sido desarrollado mediante los lenguajes HTML5 Y PHP, con hojas de estilos escritas en CSS3.

Para su realización, se ha partido de un template obtenido en la página web http://html5up.net/. Este template, que contiene ficheros escritos en HTML5, CSS3 y JavaScript, proveen al sitio de un diseño adaptable, lo cual era uno de los objetivos.

Tomando este template como base, se ha tenido que adaptar a las necesidades de nuestro proyecto, modificando tanto la estructura de las imágenes definida en HTML5 como los estilos definidos en los ficheros css.

Finalmente, para poder acceder a datos alojados en la base de datos, se ha tenido que incluir parte de código PHP que cumpliera esa función.

El sitio web está compuesto por los siguientes elementos:

- **Index.html:** se trata de la página principal del sitio web. Contiene información acerca del recorrido y de la aplicación, así como un enlace a su descarga. También contiene una lista desplegable con enlaces a las páginas de cada uno de los árboles. Como podemos ver en las siguientes figuras, la página es adaptable a los diferentes tamaños de pantalla.

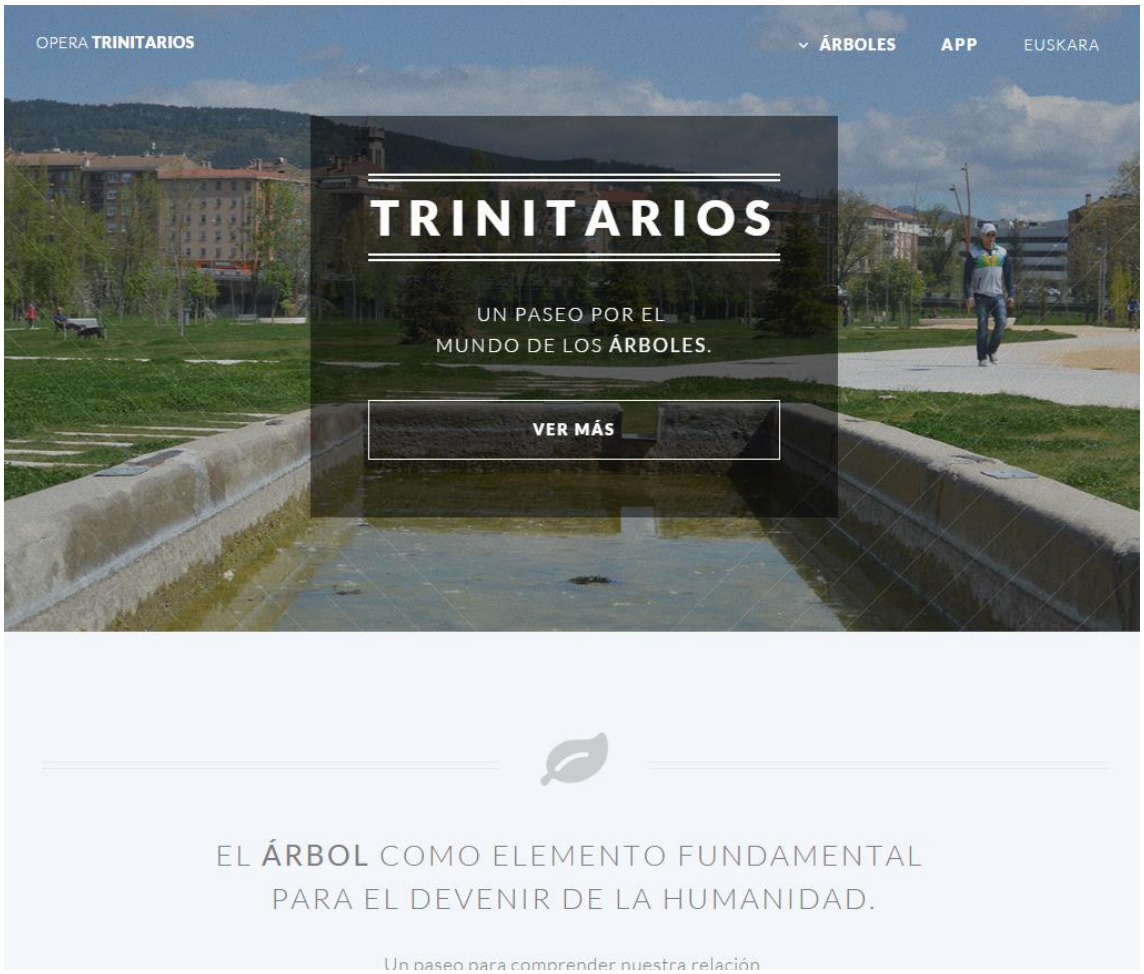



**Figura 5.10 Vista de la página principal en versión de escritorio**

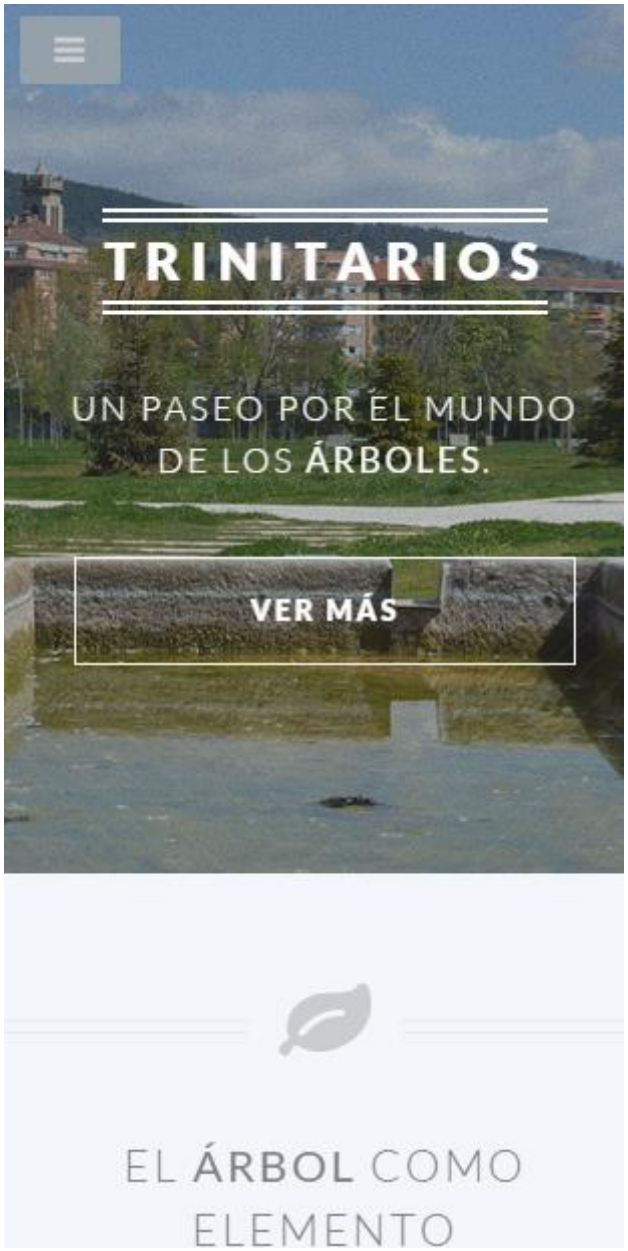



**Figura 5.11 Vista de la página principal en versión móvil**

- **xxxxxx_yyyyy.php:** se trata de 14 ficheros en total, uno por cada árbol del recorrido, donde xxxxxx_yyyyy es el id del elemento. Muestra la información del árbol, así como una imagen del mismo, y también la posibilidad de escuchar el audio de ese árbol. Esta página tiene que consultar en la base de datos para obtener la información del árbol y poder así mostrarla. En la siguiente figura vemos cómo obtenemos el objeto que necesitamos haciendo uso del fichero DBHandler, y cómo obtenemos uno de los campos del objeto para determinar, en este caso, el título de la página.

```
<!DOCTYPE HTML>
<html>

    <?php

        header("Cache-Control: public, max-age=3600");

        include $_SERVER ['DOCUMENT_ROOT'] . "/trinitarios/libs/DBHandler.php";

        $sql = "SELECT * FROM element_spa WHERE id = 'abies_nordmanniana'";
        $result = getObjectSQL ( $sql );
        $mTree = $result[0];


    ?>

    <head>
        <title><? echo $mTree['name']; ?></title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <!--[if lte IE 8]><script src="assets/js/ie/html5shiv.js"></script><![endif]-->
        <link rel="stylesheet" href="assets/css/main.css" />
        <!--[if lte IE 8]><link rel="stylesheet" href="assets/css/ie8.css" /><![endif]-->
        <!--[if lte IE 9]><link rel="stylesheet" href="assets/css/ie9.css" /><![endif]-->
    </head>
    <body class="no-sidebar">
        <div id="page-wrapper">
```

**Figura 5.12 Fichero abies_nordmanniana.php**

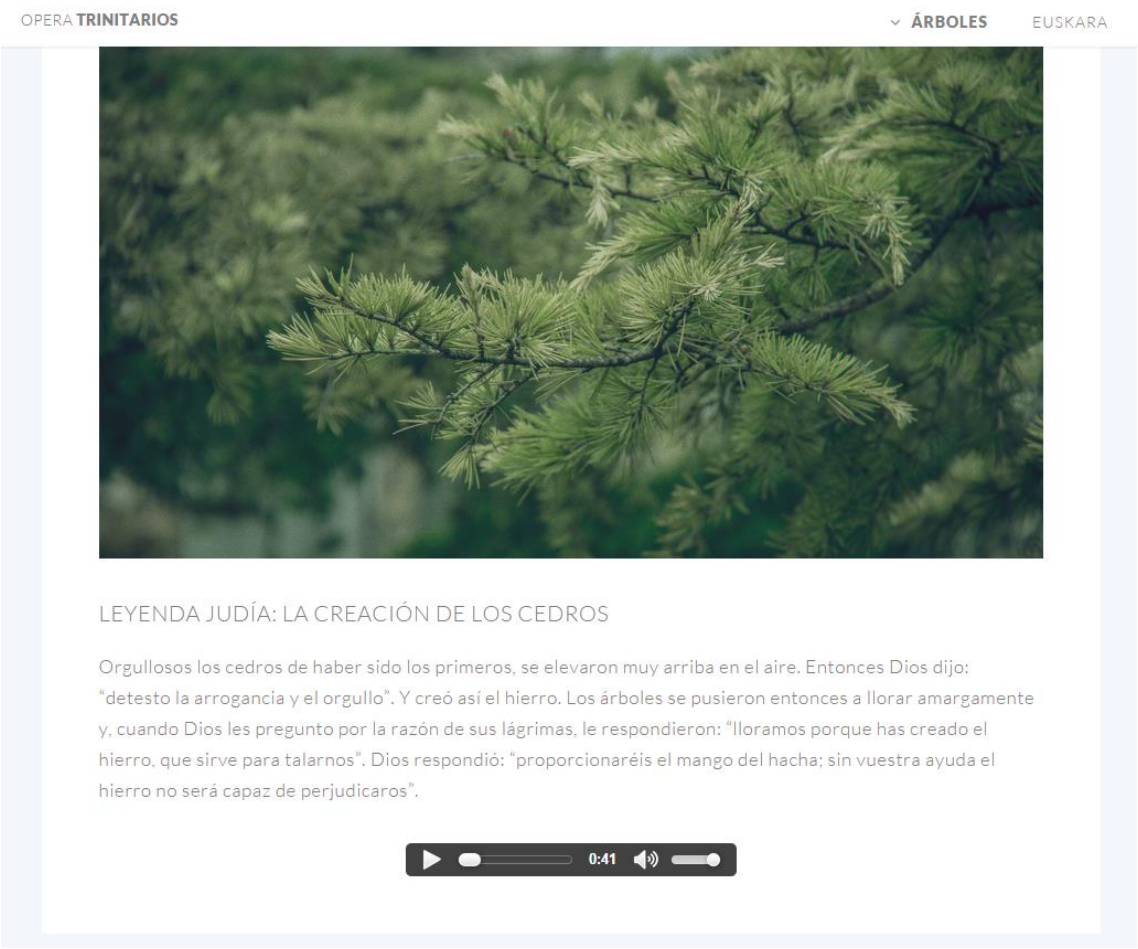



**Figura 5.13 Vista de la página de un árbol en versión de escritorio**


**Figura 5.13 Vista de la página de un árbol en versión móvil**

- **Carpeta assets**: Contiene todos los ficheros utilizados para aportar estilos, fuentes y animaciones a las páginas del sitio web. Entre los ficheros que se encuentran en esta carpeta, el más importante es main.css, donde se definen los estilos que se utilizan en todas las páginas del sitio.

- **Carpeta eus:** Contiene las páginas del sitio que están es euskera. Su contenido y funcionamiento es el mismo que la parte de castellano, pero cambiando lógicamente las rutas a los audios y las consultas a la base de datos.

# 6 PRUEBAS Y RESULTADOS

A lo largo de todo el transcurso del proyecto se han ido probando los progresos realizados, lo cual ha permitido ir detectando posibles fallos, funcionamientos no deseados y cosas a mejorar. Para estas pruebas, se han utilizado distintos dispositivos, procurando abarcar un abanico lo más amplio posible, tanto en tamaños, versiones de Android o prestaciones. Algunos de estos dispositivos han sido:

- **Samsung Galaxy Ace II**:
  - 480 x 800 píxeles
  - 3,8 pulgadas
  - Android 4.1.2
  - 768 MB de RAM
  - Procesador dual-core 800 MHz

- **Samsung Galaxy S II**:
  - 480 x 800 píxeles
  - 4,3 pulgadas
  - Android 4.2.2
  - 1 GB de RAM
  - Procesador dual-core 1,2 GHz

- **HTC One**:
  - 1080 x 1920 píxeles
  - 4,7 pulgadas
  - Android 5.0.2
  - 2 GB de RAM
  - Procesador quad-core 1,7 GHz

- **Samsung Galaxy Note 8.0**:
  - 720 x 1280 píxeles
  - 8 pulgadas
  - Android 4.1.2
  - 2 GB de RAM
  - Procesador quad-core 1,6 GHz

- **Samsung Galaxy Tab 10.1**:
  - 800 x 1280 píxeles
  - 10.1 pulgadas
  - Android 4.0.3
  - 1 GB de RAM
  - Procesador dual-core 1 GHz

## 6.1 PRUEBAS DEL SERVIDOR

Para ir probando todas las funcionalidades del lado del servidor de nuestro proyecto, se instaló el programa XAMPP. Este programa de software libre consiste en un sistema de gestión de bases de datos MySQL, un servidor Apache e intérpretes de lenguajes como PHP, entre otros.

De esta manera, gracias a este programa la modificación de los ficheros y configuración de nuestro servidor se realizaba primero en una máquina local, mediante la que se hacían todas las pruebas necesarias de una forma más rápida y sencilla, sin tener que estar continuamente subiendo nuevos ficheros a nuestro servidor alojado en el hosting contratado.

La instalación y configuración del programa es muy sencilla y, al basarse en MySQL y permitir ficheros PHP, ha sido posible replicar fácilmente todo el servidor local al externo una vez realizadas las pruebas correspondientes.

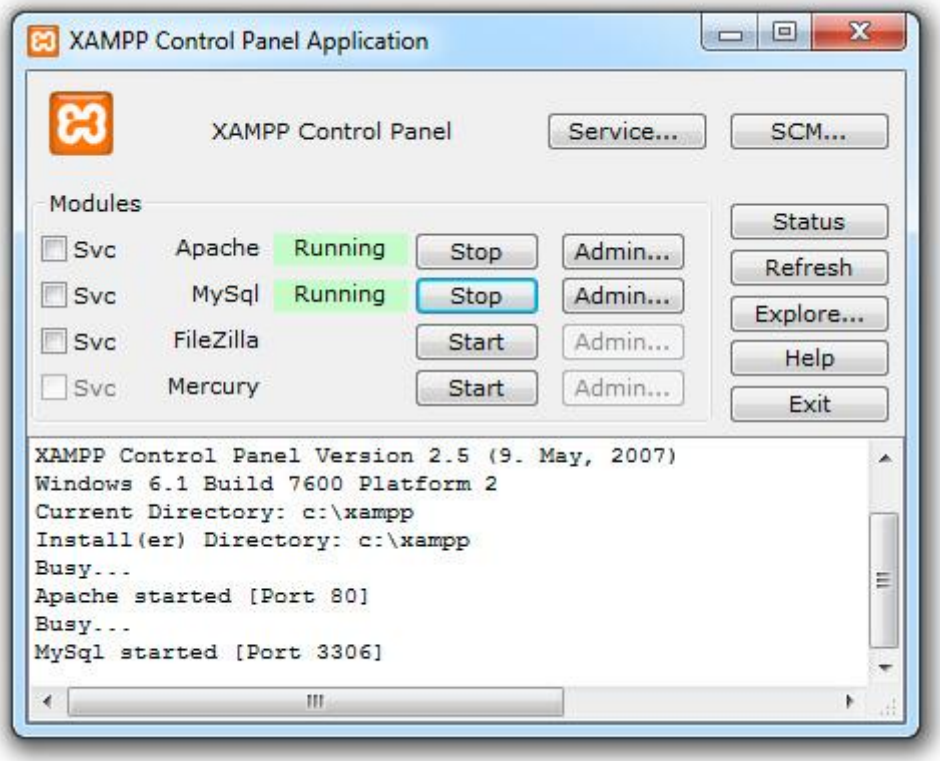



**Figura 6.1 Panel de control de XAMPP**

## 6.2 PRUEBAS DE LOCALIZACIÓN

A la hora de desarrollar una aplicación como la que se ha realizado, en la cual se hace uso de una funcionalidad como es la geolocalización, resulta complicado probar su correcto funcionamiento, ya que no es viable estar desplazándote al lugar donde supuestamente va a estar el usuario (en este caso, el parque de Trinitarios).

Para poder simular la localización del dispositivo en otra zona distinta, Android dispone de una funcionalidad llamada "mock locations" que permite a las aplicaciones enviar localizaciones falsas que son recogidas por el sistema operativo y tratadas como si de una actualización de localización real se tratara. Para ello, debemos tener activada la casilla de "Ubicaciones simuladas" en las opciones de desarrollador de nuestro dispositivo y utilizar una aplicación que haga uso de esta funcionalidad.

En este caso, se ha utilizado la aplicación Mock Locations que, como se ve en la figura 6.2, permite dibujar una ruta y simular de esta manera un recorrido por el parque.

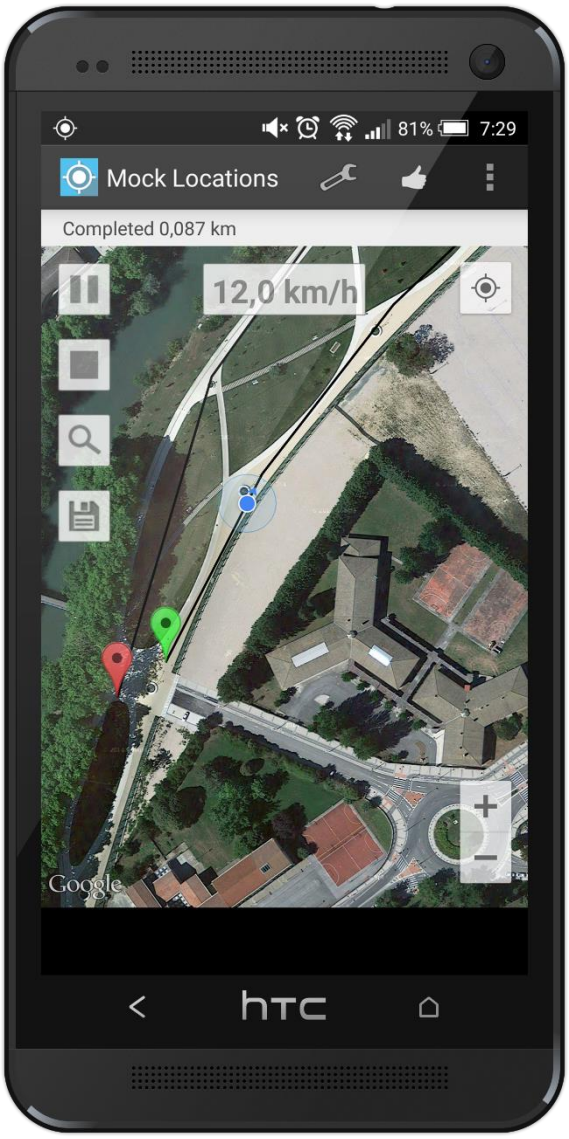



**Figura 6.2 Aplicación Mock Locations**

## 6.3 PROBLEMAS SURGIDOS

A raíz de las pruebas realizadas, ha habido partes del proyecto que se han replanteado o se han cambiado para conseguir solucionar los fallos surgidos o para simplemente mejorar el funcionamiento de algunos aspectos. A continuación se describe uno de los problemas más importantes aparecidos a lo largo del proyecto, y la solución adoptada.

- **Mapa del parque**:

Al momento de comenzar el proyecto, las imágenes de satélite proporcionadas por la API de Google Maps para Android no estaban actualizadas y mostraban el parque en un estado previo a su construcción, por lo que resultaba totalmente irreconocible y, por lo tanto, inútil para nuestra aplicación. Más tarde, las nuevas tomas del satélite con el parque ya acabado, llegaron a la versión web de Google Maps. Sin embargo, en Android seguían sin estar actualizadas. Por lo tanto, y a la espera de esa actualización, se tuvo que optar por una solución temporal al problema.

Se optó por añadir una imagen del parque encima del propio mapa de Google Maps, haciendo uso del método addGroundOverlay, que coloca la imagen que nosotros queramos, seleccionando un punto de ancla y una distancia en metros para la altura y la anchura. Este método suponía un problema añadido, ya que la imagen a superponer debía tener la máxima resolución posible, ya que al hacer zoom en el mapa, la imagen también se amplía, lógicamente, y se quería evitar que se viese muy pixelada. Esto entraba en conflicto directamente con la capacidad de memoria de los dispositivos, ya que la carga de imágenes de mayor resolución que la propia pantalla del dispositivo hace que la memoria disponible se agote rápidamente, generando una excepción del tipo OutOfMemoryError y deteniendo la aplicación.

Por ello, se tuvieron que crear tres versiones del mapa: una de 4096 x 4096 píxeles, otra de 2048 x 2048, y una última de 1024 x 1024. El que las imágenes sean potencia de 2 hace que sea más manejable por parte del sistema operativo.

De esta forma, a la hora de colocar nuestra imagen encima del mapa, lo que se hacía era intentar añadir la de máxima resolución y, si desbordaba la memoria disponible, optar por la siguiente imagen. En ninguno de los dispositivos probados se dio el caso de que la imagen de menor resolución (la de 1024 x 1024) desbordara la memoria, por lo que se optó por dejar esta imagen como la última opción.

Este algoritmo se puede ver en la siguiente figura.

```java
/**
 * Add the image of the park as an overlay over the google map.
 * There are three images with different sizes, so if one of them is too big for the
 * device's available memory, we can go for a smaller one.
 */
private void addOverlay() {

    int[] mapResources = {R.drawable.map_4096, R.drawable.map_2048, R.drawable.map_1024};
    boolean retry = true;
    int i = 0;

    // Provide an image that doesn't throw an OOM error.
    while (retry && i < 3) {
        try {
            retry = false;
            // Add the image to the map.
            mMap.addGroundOverlay(new GroundOverlayOptions()
                    .image(BitmapDescriptorFactory.fromResource(mapResources[i]))
                    .anchor(0, 1)
                    .position(MAP_ANCHOR, 700f, 700f)
                    .zIndex(0));
        } catch (Exception e) {
            e.printStackTrace();
        } catch (OutOfMemoryError e) {
            i++;
            retry = true;
            e.printStackTrace();
        }
    }
}
```

**Figura 6.3 Método addOverlay**

Finalmente las imágenes de Google Maps fueron actualizadas y mostraban ya el parque en su estado actual, por lo que no fue necesario hacer uso de estas imágenes ni de este método.

## 6.4 PRUEBA FINAL IN SITU

Con el fin de hacer una valoración del resultado final de este proyecto, se ideó una prueba con usuarios reales en el propio parque, simulando lo que sería una visita real al mismo. Estos "visitantes" recibieron una serie de pautas a seguir a la hora de hacer el recorrido, ya que se deseaba comprobar el correcto funcionamiento de todas las características en todos los dispositivos.

Para realizar la prueba, y ante la imposibilidad de instalar placas metálicas ancladas al suelo debido a la falta de un permiso del ayuntamiento, lo que se hizo fue colocar las placas (impresas en cartón y plastificadas) sobre unos soportes con forma de trípode, como puede verse en las fotografías tomadas durante la prueba.

La prueba contó con la participación de un total de 12 personas de edades comprendidas entre los 13 y 57 años.

Las instrucciones que recibieron los participantes fueron:

1. Prueba a descargarte la aplicación Opera Trinitarios. Puedes hacerlo escaneando el código QR que hay en las placas colocadas en las entradas al parque o directamente buscándola en Google Play.
2. Si no has podido instalar la aplicación, prueba a realizar el recorrido escaneando los códigos QR de las placas y viendo la información por medio de páginas web.
3. Si has podido instalar la aplicación:
    a. Prueba a realizar el recorrido escaneando los códigos QR de las placas con la aplicación y visualizando la información desde la propia aplicación.
    b. Prueba a realizar el recorrido mediante la audio-guía. Fíjate si los audios suenan cuando deben y si el audio corresponde con el árbol del que te encuentras cerca en ese momento.
    c. Navega libremente por la aplicación, cambia el idioma, etc. tratando de comprobar si todo funciona correctamente y cómo esperarías que lo hiciera.

La prueba se realizó en el parque de Trinitarios y duró aproximadamente dos horas. A continuación se muestras algunas fotografías tomadas durante la misma.


ASTIGAR
Acer Pseudoplatanus
ARCE
Acer Pseudoplatanus
upna
OFERA

## 6.5 RESULTADOS

Para visualizar los resultados de la prueba, se realizó el siguiente formulario, que fue distribuido a los participantes para su cumplimentación:

Enlace al formulario:

https://docs.google.com/forms/d/1_ggvbCQhjJidNRHkWZdTktjLEfa2qUrrYjw7tEmlfGY/viewform?usp=send_form

En base a las respuestas recibidas, se extraen los siguientes resultados de la prueba:

- 10 de las 12 personas pudieron instalar la aplicación y recorrer el parque haciendo uso de ella. Las otras 2 personas pudieron realizarlo mediante el sitio web.

- 3 de las 10 personas utilizaron la aplicación en euskera. Las otras 7 lo hicieron en castellano.

- Dos de los códigos QR no se correspondían con el árbol. Se atribuye este error a un fallo a la hora de copiar los códigos QR en las placas.

- 9 de las 10 personas afirman que la audio-guía ha funcionado correctamente, aunque la mayoría han tenido que esperar un poco hasta que la señal de GPS se ha estabilizado para poder empezar el recorrido. La otra persona alega que su señal de GPS era muy poco estable y, por lo tanto, los audios no siempre saltaban cuando debían, sino que a veces saltaban cuando se encontraba ya lejos del árbol correspondiente. Esta circunstancia se atribuye a que el dispositivo que portaba el participante era de gama más baja y probablemente su chip detector de GPS no tenía la precisión necesaria.

- 2 de las personas que visitaron la web no pudieron escuchar los audios.

- Se recogen algunas sugerencias de cara a mejorar la experiencia:

    - Enlazar desde la aplicación a otras páginas webs para ampliar la información de los árboles.
    - La posibilidad de que la opción de cambiar de idioma esté más a la vista, como en la barra de acción de la aplicación.

Además, mientras se realiza la prueba, otros visitantes ajenos a la misma se paran a leer los carteles, constatando la curiosidad que suscitan este tipo de actuaciones.

En general, el resultado de la prueba es muy positivo, ya que no se han detectado grandes fallos, más allá de los comentados. La totalidad de los participantes afirma haber disfrutado del recorrido, aplaude la idea y anima a seguir adelante con este tipo de iniciativas.

# 7 CONCLUSIONES Y LINEAS FUTURAS

En este apartado se van a resumir los objetivos completados a lo largo de este tiempo, así como las conclusiones personales y las líneas futuras del proyecto.

## 7.1 OBJETIVOS CUMPLIDOS

Al término de este proyecto, podemos decir que se ha conseguido:

- Se ha conseguido recopilar y almacenar en un servidor toda la información que se ha considerado necesaria acerca de los árboles seleccionados para formar parte del recorrido interactivo.
- Se ha realizado una aplicación para el sistema operativo Android que permite recorrer el parque de una manera interactiva, amena y, al fin y al cabo, diferente. Esta aplicación cumple con todos los objetivos y requisitos que se habían marcado y además, ha sido realizada poniendo un especial cuidado en que tanto su rendimiento sea el más rápido y fluido posible, como en que su diseño se aproxime lo máximo posible a las pautas marcadas por Google con Material Design.
- Se ha realizado un pequeño sitio web que muestra toda la información de una manera clara y sencilla y que sirve al mismo tiempo de página de aterrizaje para los usuarios que no conozcan ni el recorrido ni la aplicación. Se ha conseguido el objetivo de que este sitio web fuera totalmente adaptable, de manera que se visualice todo el contenido de una manera correcta y clara independientemente del dispositivo utilizado.

## 7.2 CONCLUSIONES

Las conclusiones son todas muy positivas. Creemos que partiendo de una idea que ya de por sí resultaba muy interesante, se ha conseguido un resultado más que aceptable, cumpliendo todos los objetivos y constatando que el resultado es del agrado del usuario final.

Personalmente, durante la realización de este proyecto considero que he adquirido unos conocimientos y una experiencia que me puede resultar realmente útil a la hora de enfocar mi futuro en este ámbito:

- He aprendido a utilizar un IDE como Android Studio, que es el IDE de referencia actualmente en el desarrollo de aplicaciones Android, y el cual desconocía anteriormente.
- He ampliado notablemente mis conocimientos de desarrollo en Java, y más concretamente de desarrollo Android.

- He adquirido nuevos conocimientos sobre las nuevas formas de desarrollo web, y más concretamente he aprendido cómo realizar una página web que sea adaptable, haciendo uso tanto de HTML5 como de hojas de estilos CSS3.

Además de este tipo de conocimientos y aptitudes logradas, quiero destacar el haber podido experimentar lo que es trabajar en equipo, aunque fuese un equipo pequeño, ya que me ha hecho ser consciente de la importancia de poner continuamente asuntos en común para poder debatirlos, llegar a soluciones consensuadas y avanzar de esa manera en el proyecto.

## 7.3 LÍNEAS FUTURAS

A pesar de que los objetivos marcados para el proyecto se han cumplido, y que los resultados son muy satisfactorios, también hemos constatado por medio del feedback recibido en la prueba realizada que hay muchas cosas que pueden mejorarse:

- Implementar un modo offline: aunque sabemos que sin conexión no podríamos hacer uso ni del mapa ni de la audio-guía, podría resultar interesante que por lo menos se pudiese visualizar la lista de elementos y su información, descargándola previamente, para poder de esta manera visualizarla aunque no se disponga de acceso a internet.
- Traducir el contenido al inglés: sería interesante que, así como la aplicación, el contenido también estuviera disponible en inglés (incluidos los audios), para que aquellos visitantes extranjeros que vengan al parque puedan disfrutar del recorrido.
- Enriquecer el contenido: una de las cosas que reclamaban algunos usuarios era que el contenido mostrado tanto en la aplicación como en la web fuese más rico y variado.
- Aunque en la realización de la audio-guía se ha procurado encontrar un equilibrio entre rendimiento y consumo de batería, creemos que este apartado todavía se podría pulir mucho más, buscando quizás una estrategia en la geolocalización que permita utilizar el GPS sólo cuando realmente sea necesario, y reducir lo máximo posible el consumo de batería.

Además de estas mejoras, creemos que este proyecto está abierto a muchísimos posibles enfoques que enriquezcan y complementen la idea original de servir como fuente de conocimiento y concienciación con nuestro entorno. En ese sentido, una de las posibles actuaciones futuras sería enfocar la aplicación a un uso educativo por parte de los centros, de manera que los niños y niñas, atraídos por el uso de las nuevas tecnologías, sean capaces de hacer uso de ellas para formarse, educarse, y fomentar entre ellos valores como el amor por la naturaleza o la importancia del legado cultural de nuestros antepasados.

Para conseguir esto, una de las ideas sería aportar a la aplicación un componente lúdico, en forma de alguna especie de gymkana en la que los chavales y chavalas puedan ir acompañados de su dispositivo y superando diferentes pruebas en las que tengan, por ejemplo,

que descubrir a qué árbol corresponde una determinada hoja, o tengan que ir descubriendo historias a través de audios que les vayan guiando hacia una meta determinada, etc.

Consideramos, en definitiva, que el resultado de este proyecto puede servir como base para actuaciones mucho más integrales en el propio parque, y ser utilizado al mismo tiempo como prototipo para actuaciones similares en otros parques, conjuntos históricos y turísticos e incluso ciudades enteras.

# 8 BIBLIOGRAFÍA

## 8.1 CONTENIDO

Para la creación del contenido, se han consultado:


- ABELLA MINA, I. (1996). *La magia de los árboles.* RBA Libros.
- ABELLA MINA, I. (2012). *El gran árbol de la humanidad.* RBA Libros.
- CHANES, R. (2009). *Deodendron.* Blume.
- LÓPEZ GONZÁLEZ, G. (2007). *Guía de los árboles y arbustos de la Península Ibérica y Baleares.* S. A. Mundi-Prensa Libros.
- NEWMAN, A. (2009). *Árboles; guardianes de la magia.* Oceano Ambar.
- WILKINSON, J. y MITCHELL, A. (1999). *Pequeño manual de los árboles de Europa.* Omega.

- *Guías de patrimonio natural de Navarra.*


## 8.2 IMÁGENES

Las imágenes han sido obtenidas de la red social Flickr, dedicada a la fotografía. Todas las imágenes utilizadas están bajo licencia Creative Commons, estando permitida su utilización para fines no comerciales.

- **Flickr**:
  https://www.flickr.com/

## 8.3 ICONOS

Para la realización de este proyecto se han utilizado una serie de iconos, tanto para la aplicación como para el sitio web.

- **Aplicación Opera Trinitarios**:

  Se han utilizado los iconos disponibles en la siguiente web:
  https://materialdesignicons.com/

  Licencia Creative Commons:

  https://github.com/google/material-design-icons/blob/master/LICENSE

- **Sitio web**:

  Se han utilizado los iconos disponibles en la siguiente web: http://fortawesome.github.io/Font-Awesome/icons/

  Licencia SIL OFL 1.1: http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL

## 8.4 LIBRERÍAS

En la aplicación realizada hemos hecho uso de algunas librerías externas de software libre que nos han permitido disponer de ciertas funcionalidades y ciertos aspectos estéticos que se han considerado necesarios para la realización de la aplicación.

- **Picasso**: librería que permite un rápido y eficaz manejo y carga de imágenes en la aplicación Android: http://square.github.io/picasso/ (Apache License 2.0)

- **BarcodeScanner ZBar**: librería que provee las clases y métodos necesarios para el lector QR de nuestra aplicación. Está basada a su vez en el software de código abierto Zbar: https://github.com/dm77/barcodescanner (Apache License 2.0)

- **SmoothProgressBar**: pequeña librería que nos ha permitido mostrar una barra de progreso circular indeterminada en nuestra aplicación: https://github.com/castorflex/SmoothProgressBar (Apache License 2.0)

- **FloatingActionButton**: librería que permite mostrar un FloatingActionButton como el que mostramos en la sección de audio-guía de nuestra aplicación: https://github.com/makovkastar/FloatingActionButton (The MIT License)

## 8.5 TEMPLATE

Para el diseño de nuestro sitio web, se ha utilizado un template para HTML5 y CSS3 que posteriormente se ha modificado y adaptado a las necesidades del proyecto.

El template es el siguiente: http://html5up.net/twenty (Licencia Creative Commons Attribution 3.0)

## 8.6 REFERENCIAS

A continuación se muestran los enlaces que se han utilizado para extraer información a la hora de realizar el proyecto:

- Conocimientos de desarrollo para Android:
  http://developer.android.com/index.html

- Información acerca de Material Design:
  http://www.google.com/design/

- Información acerca de la API de Google Maps para Android:
  https://developers.google.com/maps/documentation/android/?hl=es

- Conocimientos básicos de Adobe Illustrator:
  http://tv.adobe.com/es/show/aprende-illustrator-cc/

- Conocimientos de desarrollo para HTML5 y CSS3:
  http://www.w3schools.com/

# 9 ANEXO I: PLACAS INFORMATIVAS

En este apartado se encuentra una pequeña muestra de las placas que se han instalado en el parque. Como las placas de los distintos árboles difieren únicamente en la información mostrada, se ha seleccionado sólo una de ellas como muestra. Además, se muestra la placa ideada para colocarse en las distintas entradas del parque, a modo de reclamo y bienvenida al recorrido. Como se puede observar, las placas están en castellano y euskera, y contienen además códigos QR que pueden usarse para visitar el sitio web o junto con la aplicación para ver la información del elemento correspondiente.


**Figura 10.1 Placa de entrada en castellano**


**Figura 10.2 Placa de entrada en euskera**


**Figura 10.3 Placa de árbol en castellano**


**Figura 10.4 Placa de árbol en euskera**